Voorwaartse en achterwaartse surrogaatmodellering
van computationeel dure problemen

Forward and Inverse Surrogate Modeling
of Computationally Expensive Problems

Ivo Couckuyt

Promotoren: prof. dr. ir. T. Dhaene, prof. dr. ir. F. De Turck
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. D. De Zutter
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2012 - 2013

UNIVERSITEIT
GENT

ISBN 978-90-8578-599-6
NUR 910, 980
Wettelijk depot: D/2013/10.500/32

Dankwoord

Een boek gaat over wat je er zelf uithaalt. Niet over wat de schrijver belangrijk vindt.
— Thea Beckman

Na een 5 jaar lange tocht loopt mijn doctoraat ten einde. Ik zou het nooit gehaald hebben zonder de hulp, het gezelschap en de steun van vrienden en collega's. Ik wil dan ook een aantal mensen bedanken die belangrijk zijn geweest voor mij en voor het behalen van dit doctoraat. Op professioneel vlak wil ik allereerst natuurlijk mijn promotor Tom Dhaene bedanken voor de kans om dit onderzoek uit te voeren en me alle vrijheid te geven om me volledig te concentreren op mijn werk. Daarnaast wil ik mijn (ex-)collega's Dirk Gorissen, Karel Crombecq, Dirk Deschrijver en Minh Nguyen bedanken voor de vele waardevolle discussies en voor hun medewerking aan verscheidene artikels. Het was een zeer leuke ervaring om samen te werken aan het grootschalig software project dat we van SUMO gemaakt hebben. Zonder hun aanwezigheid zou de periode van 5 jaar maar moeilijk door te komen geweest zijn. Over deze laatste 5 jaar heb ik ook verscheidene nieuwe kennissen gemaakt over de hele wereld (waarvan ik sommige ook al vrienden zou noemen) als gevolg van verscheidene samenwerkingen. Deze connecties hebben mijn inzicht in deze wereld en zijn verscheidene culturen grondig verrijkt en verruimd. Op persoonlijk vlak wil ik ook mijn familie bedanken, die ondertussen al enkele leden groter is geworden, voor hun steun en vertrouwen die me uiteindelijk tot hier hebben geleid. Alhoewel onderzoek een groot deel van een dag opvult van een doctoraatsstudent blijft er natuurlijk vrije tijd over die ik doorbracht met de vrienden. Hun wil ik bedanken voor de vele toffe (café, film, etc.) avonden waar ik mijn gedachten eens vrij kon maken. Als laatste wil ik vanzelfsprekend ook iedereen bedanken die me de voorbije jaren geholpen hebben en die ik hier vergeten ben te vermelden. Mocht dit zo zijn dan is dit een fout van mijn kant. Jullie vriendschap en steun wordt zeker geapprecieerd!

Antwerpen, februari 2013
Ivo Couckuyt

Contents

List of Figures

List of Tables

List of Abbreviations

AAE Average absolute error

AEE Average euclidean error

AGPL Affero GNU general public license

AIC Akaike information criterion

ANN Artificial neural network

ARE Average relative error

BEEQ Bayesian estimation error quotient

BLUP Best linear unbiased predictor

CAD Computer-aided design

CAE Computer-aided engineering

CFD Computational fluid dynamics

CST MWS CST microwave studio

cvpe Cross validated prediction error

DACE Design and analysis of computer experiments

DIRECT Dividing rectangles

DoE Design of experiments

EGO Efficient global optimization

EI Expected improvement

EM Electromagnetics

EMO Efficient multiobjective optimization

EMS Evolutionary model selection

FEBio Finite elements for bio-mechanics

GA Genetic algorithm

GAE Geometric average error

GIS Geographical information system

GP Gaussian process

gPoI Generalized probability of improvement

GRE Geometric relative error

GRFM Gaussian random field metamodels

HAE Harmonic average error

HDMR High dimensional model representation

HPC High performance computing

HRE Harmonic relative error

HSO Hypervolume by slicing objectives

LHD Latin hypercube design

LNA Low noise amplifier

LOO Leave-one-out

LS-SVM Least squares support vector machine

LVDT Linear variable differential transformer

MAE Maximum absolute error

MAO Metamodel-assisted optimization

MD Minimum distance

MLE Maximum likelihood estimation

MOEA Multiobjective evolutionary algorithm

MOSBO Multiobjective surrogate-based optimization

MRE Maximum relative error

MSE Mean squared error

NSGA-II Non-dominated sorting genetic algorithm II

ooDACE toolbox Object-oriented DACE toolbox

ParEGO Pareto efficient global optimization

PDF probability density function

PoI Probability of improvement

PSO Particle swarm optimization

QOR Quasi-optimal region

RAEI Relative absolute error I

RAEII Relative absolute error II

RBF Radial basis functions

RMSE Root mean square error

RMSRE Root mean square relative error

RRSE Root relative square error

RSE Relative squared error

SBO Surrogate-based optimization

SMS-EMOA S-metric selection evolutionary multiobjective algorithm

SPEA2 Strength Pareto evolutionary algorithm 2

SQP Sequential quadratic programming

SUMO toolbox Surrogate modeling toolbox

SVM Support vector machine

SVR Support vector regression

WFG Walking fish group

Samenvatting

Voor veel wetenschappelijke (ingenieurs-) problemen is het niet praktisch om fysische experimenten rechtstreeks uit te voeren. In de plaats daarvan worden er complexe computersimulaties gebruikt, om zo de kosten te drukken of het gevaar te minimaliseren. Deze computersimulaties worden door de ingenieur gebruikt om het gedrag te bestuderen van het systeem. Computersimulaties zijn dikwijls geparametriseerd door een set van input parameters (variabelen) en een set van output parameters (responses). Gegeven een set van inputs zal het simulatiemodel de corresponderende outputs berekenen. Het bestuderen van zulke input-output systemen bestaat erin de input parameters te wijzigen en het gevolg daarvan op de outputs te observeren. Het doel is om de volledige relatie tussen de verscheidene input parameters (ontwerp parameters) en de output parameters (performantie karakteristieken) te begrijpen. Deze virtuele experimenten bieden wetenschappers meer flexibiliteit om fenomenen te bestuderen in gecontroleerde omstandigheden. Computersimulaties vereisen echter vaak ook een aanzienlijke investering in tijd en rekenkracht: één enkele computersimulatie kan enkele minuten, uren, dagen of zelfs weken in beslag nemen. Bijvoorbeeld, Ford Motor Company deelde mee dat een volledige crash simulatie van een passagiers auto 160 uren kan duren. Hierdoor is het rechtstreeks gebruik van computersimulaties moeilijk of zelfs onmogelijk voor ingenieurs die het systeem willen bestuderen, optimaliseren, enz. Daarom gebruiken wetenschappers meer en meer goedkopere approximatie technieken, ook bekend als surrogaatmodellen, die het gedrag van de computationeel dure computersimulatie zo goed mogelijk nabootst.

Het doel van globale surrogaatmodellering is het ontwerpen van een surrogaatmodel dat zo accuraat mogelijk is, met zo weinig mogelijk computersimulaties. Een surrogaatmodel wordt geconstrueerd door het uitvoeren van meerdere simulaties op belangrijke locaties in de input ruimte waarna een surrogaatmodel wordt geselecteerd dat de data en het gedrag van het volledige systeem zo goed mogelijk benadert. Nadat een surrogaatmodel beschikbaar is kan het gebruikt worden om de computersimulatie te vervangen in verder onderzoek. Surrogaatmodellering is heel waardevol voor talloze onderzoeksdomeinen, van werktuigbouwkunde tot hydrologie. Het creëren van een surrogaatmodel is echter een volledig onderzoeksdomein op zich en talloze opties zijn ter beschikking voor de ontwerper. Deze opties omvatten de keuze van het type surrogaatmodel, distributie van de data-

punten, optimalisatiestrategieën voor parameters van het surrogaatmodel, enz. Het is intuïtief duidelijk dat de distributie van datapunten cruciaal is bij het creëren van een accuraat surrogaatmodel. Traditioneel worden alle datapunten gekozen en gesimuleerd a priori, en een surrogaatmodel wordt enkel gecreëerd nadat alle datapunten geëvalueerd en beschikbaar zijn. In een sequentieel ontwerp worden datapunten iteratief geselecteerd, intermediaire surrogaatmodellen en de vorige datapunten worden gebruikt om een betere keuze te maken van toekomstige experimenten.

Dit werk legt de focus op het gebruik van data gebaseerde, globale surrogaatmodellen om veel voorkomende taken te versnellen zoals optimalisatie, en het exploreren van de ontwerpruimte. In dit geval is een surrogaatmodel slechts een hulpmiddel om de vooropgestelde taak te voltooien met zo weinig mogelijk computationeel dure simulaties. Achteraf wordt het surrogaatmodel vaak weggegooid omdat het meestal onvoldoende accuraat is om de dure computersimulatie volledig te vervangen. Om het voordeel van surrogaatmodellering zoveel mogelijk te exploiteren is het nodig om het gebruik van surrogaatmodellen zoveel mogelijk te integreren in het (optimalisatie) proces. Het domein van dit proefschrift bevindt zich in de intersectie van verschillende onderzoeksdomeinen zoals machinaal leren (artificiële intelligentie), gedistribueerde systemen, modellering & simulatie, en software engineering. Al de methoden gepresenteerd in dit proefschrift zijn vrij beschikbaar als deel van de SUrrogate MOdeling (SUMO) en ooDACE software pakketten. SUMO is een platform geschreven in Matlab voor adaptieve surrogaatmodellering met sequentieel ontwerp dat het volledige modelleringsprocess aanpakt, van het evalueren van de eerste datapunten tot het trainen van surrogaatmodellen en optimalisatie. ooDACE is een collectie van populaire Kriging surrogaatmodel types in één software pakket die gemakkelijk kan geïntegreerd worden in bestaande code. Beide pakketten zijn open-source, and dus alle resultaten van dit proefschrift kunnen gereproduceerd en gevalideerd worden. SUMO en ooDACE zijn beschikbaar via `http://sumo.intec.ugent.be`.

Het eerste deel van Hoofdstuk 3 geeft een overzicht van globale surrogaatmodellering en de problemen die ermee gepaard gaan. In het bijzonder wordt de creatie van surrogaatmodellen steunend op meerdere modelselectiecriteria bestudeerd. Het tweede deel van Hoofdstuk 3 introduceert het ooDACE software pakket wat de basis is van verscheidene (optimalisatie) algoritmes van dit proefschrift. Deze efficiënte Kriging implementatie wordt gebruikt door onderzoekers over de hele wereld. Een nieuwe veelbelovende Kriging techniek, blind Kriging, wordt uitgebreid getest op een uiteenlopende set van relevante problemen.

Bekende optimalisatiemethoden gebaseerd op het Kriging surrogaatmodel vormen de basis van dit proefschrift. Hoofdstuk 3 geeft een overzicht van allerlei surrogaatmodel gebaseerde optimalisatiemethoden en werkt twee toepassingen uit met behulp van het Efficient Global Optimization algoritme (EGO). Het EGO algoritme wordt gebruikt om zowel voorwaartse

als inverse problemen op te lossen. Het laatste deel van Hoofdstuk 3 bespreekt het gebruik van automatische surrogaatmodel selectie (met behulp van het Evolutionaire Model Selectie algoritme) voor optimalisatie. Traditioneel wordt het EGO algoritme gebruikt samen met het Kriging surrogaatmodel. EGO wordt uitgebreid om automatisch het beste surrogaatmodel type te selecteren tijdens de optimalisatie, zoals support vector regressie, radiale basis functies, enz. Het algoritme wordt getest op een optimalisatieprobleem uit de werktuigbouwkunde.

Hoofdstuk 4 behandelt efficiënte methoden om computationeel dure optimalisatieproblemen met meervoudige doelstellingen te behandelen. Het EGO algoritme wordt vaak gebruikt voor optimalisatie met één doelfunctie, maar het concept achter EGO is ook zeer geschikt voor optimalisatie met meerdere doelfuncties. In de literatuur is hier weinig werk rond verricht omdat de uitbreiding naar optimalisatie met meervoudige doelstellingen een paar computationeel moeilijkheden introduceert. Nieuwe algoritmes worden uitgewerkt die gebruikt kunnen worden om EGO efficiënt toe te passen voor optimalisatie met meervoudige doelstellingen.

Hoofdstuk 5 behandelt het gebruik van surrogaatmodellen om inverse problemen rechtstreeks op te lossen. Inverse problemen zijn niet gemakkelijk om rechtstreeks op te lossen omdat ze vaak slecht geconditioneerd zijn. De meeste gebruikte aanpak is om het inverse probleem te reduceren tot een eenvoudig voorwaarts probleem zoals het optimaliseren van de fout tussen de output van de simulatie code en een opgegeven waarde of bereik. In dit proefschrift wordt een sequentieel ontwerp ontwikkeld, gebaseerd op het Kriging surrogaatmodel, dat datapunten kiest in juist die regionen in de input ruimte waarvan de output overeenkomt met de opgegeven waarde (of bereik).

Hoofdstuk 6 bevat conclusies en toekomst perspectieven. De SUMO en ooDACE software pakketten worden meer uitgebreid besproken in Appendices A en B.

Summary

For many problems in science, and engineering it is impractical to perform experiments on the physical world directly. Instead, complex, physics-based simulation codes are used to run experiments on computer hardware. These simulation models are used by the engineer to understand and interpret the behavior of the system under study. Simulation codes are often parametrized by a set of input parameters (factors or variables) and a set of outputs (responses). Given a set of inputs, the simulation model will produce a corresponding output. Studying such input-output based systems involves changing the input parameters and observing the impact on the output of the simulation. The aim is to understand the full relationships between the different input parameters (e.g., design variables) and the outputs of the system (e.g., performance characteristics). While allowing scientists more flexibility to study phenomena under controlled conditions, computer experiments require a substantial investment of computation time. One model evaluation may take many minutes, hours, days or even weeks. For example, Ford Motor Company reported on a crash simulation for a full passenger car that takes 160 hours to compute. Because of this long computation time, using simulation models directly is still impractical for engineers who want to explore, optimize, or gain insight into the system. As a result researchers have turned to various approximation methods, also known as surrogate models, that mimic the behavior of the simulation model as closely as possible while being computationally cheap(er) to evaluate.

The goal of global surrogate modeling is the generation of a surrogate that is as accurate as possible, using as few simulation evaluations as possible. A surrogate model is constructed by performing multiple simulations (called samples) at key points in the input space, analyzing the results, and selecting a surrogate model that approximates the data and the overall system behavior quite well. Once a surrogate model is available it can replace the existing expensive simulation code further down the engineering design pipeline. Surrogate modeling is useful in a large range of scientific disciplines and fields, ranging from mechanical engineering to hydrology. However, generating a surrogate model is almost an entire research domain in itself. There are an overwhelming number of options available to the designer: different surrogate model types, different experimental designs, different surrogate model parameter optimization strategies, etc. Intuitively,

the choice of the experimental design (distribution of data points) is of paramount importance to the success of the surrogate modeling task. In traditional (one-shot) experimental design, all data points are selected and simulated a priori, and a surrogate model is trained only after all points are simulated. In sequential design, points are selected with an iterative method, in which intermediate models and outputs from previous simulations are used to make a better, more informed choice on the locations for future simulations.

This thesis concentrates on the use of data-driven, global surrogate models to expedite specific engineering tasks such as optimization, sensitivity analysis and design space exploration. In this case the surrogate model acts just as a tool to perform the required task while minimizing the number of simulations. In the end the surrogate model is often discarded as it is not necessarily globally accurate and, thus, not suitable for replacing the simulation code. To fully take advantage of surrogate modeling for optimizing the simulation code (or another engineering task), the surrogate models need to be tightly integrated into the broader process, for optimization methods this is also known as Surrogate-Based Optimization (SBO). This involves working in the intersection of machine learning/AI, distributed systems, modeling & simulation, and software engineering. All the methods proposed in this thesis are freely available in the Surrogate MOdeling (SUMO) and ooDACE toolboxes. The SUMO toolbox is a framework written in Matlab for adaptive surrogate modeling with sequential design, which tackles the entire modeling process from the first samples through simulation to model training and optimization. ooDACE is a collection of popular Kriging surrogate model types in one toolbox which can be easily integrated into an existing modeling pipeline. Both toolboxes are open source, and therefore all results in this thesis can be reproduced and validated. SUMO and ooDACE can be downloaded from `http://sumo.intec.ugent.be`.

The first part of Chapter 3 provides an in-depth overview of global surrogate modeling and associated problems. In particular, the notion of building a surrogate model using multiple accuracy measures in a multiobjective way is explored. The second part of Chapter 3 describes the ooDACE toolbox which incorporates various types of the Kriging surrogate model for the first time in one framework using object oriented programming. The ooDACE toolbox is the foundation of several (optimization) algorithms presented in this thesis. This efficient Kriging implementation has been used on its own and as part of the SUMO toolbox by researchers over the world. One type of Kriging, i.e., blind Kriging, is thoroughly benchmarked on a very distinct set of real-life problems.

Well-known SBO methods based on the popular Kriging surrogate model form the core of this thesis. Chapter 3 gives an overview of the current state-of-the-art SBO algorithms and presents two applications optimized using a popular SBO algorithm, i.e., the Efficient Global Optimization algorithm (EGO). The EGO algorithm is used to solve forward as well as

inverse problems. The remaining part of Chapter 3 discusses the use of automatic surrogate model selection (using the Evolutionary Model Selection algorithm) for optimization. Traditionally, the EGO algorithm is used in conjunction with the Kriging surrogate model. Here EGO is extended to automatically use the best surrogate model type such as support vector regression, radial basis functions, etc. and demonstrate its usefulness on an optimization problem from mechanical engineering.

Chapter 4 presents efficient methods to solve expensive multiobjective optimization problems with many objectives. The EGO algorithm is widely used for single-objective optimization though the concept is also very much suited for multiobjective optimization. Few work exists in literature as the generalization to multiobjective optimization of the EGO algorithm induces some computational difficulties. Novel algorithms are introduced that can be used to apply EGO to multiobjective optimization much more efficiently.

Chapter 5 deals with using surrogate models to solve inverse problems directly. Inverse problems are difficult to solve directly as they can be ill-conditioned. The most used approach is to reduce complex inverse problems to a more simple forward problem such as optimizing the error between the output of the simulation code and some target value or range. In this thesis a sampling strategy is developed, based on the Kriging surrogate model, that focuses on sampling the input regions densely that correspond to the desired target value or range.

Chapter 6 contains conclusions and future prospects. Finally, Appendices A and B discuss the SUMO and ooDACE toolbox in more detail, respectively.

1 Introduction

There are no facts, only interpretations.
— Friedrich Nietzsche

Scientists and engineers are often confronted with physical phenomena that are difficult to study. The ability to perform experiments directly on the physical world may be limited by the monetary cost (e.g., prototyping airfoil designs), lack of controllable and reproducible conditions (e.g., earthquake propagation, solar eclipse), etc. Instead, the last few decades complex science gradually moved into the computational domain, significantly increasing the number of problems and phenomena that can be studied. This has led to an explosion of data and tools that are at present being unified into, what is often referred to as, e-Science [17].

1.1. Problem domain

Computer simulation has been steadily replacing real-life experiments as a major tool to formulate theories and solutions. A computer simulation, also called a computer model, is a program that attempts to simulate a complex system. These simulations are used by the engineer to understand and interpret the behavior of the system. This allows scientists more flexibility to study phenomena under controlled conditions.

Simulation codes are often parametrized by a set of input parameters (factors or variables) and a set of outputs (responses). Given a set of inputs, the simulation produces a corresponding output. Studying such input-output

based systems involves changing the input parameters and observing the impact on the output of the simulation. The aim is to understand the full relationships between the different input parameters (e.g., design variables) and the outputs of the system (e.g., performance characteristics).

Computer simulations can be characterized by a number of properties, warranting the use of a different set of techniques. Moreover, certain categories of simulation codes are associated with different research domains which have their own terminology, methodology and practices. These properties are discussed in the remainder of this section.

1.1.1. Computation time

Similar to physical experiments, computer experiments might still require a substantial investment of time. This is especially evident for routine tasks such as prototyping, high dimensional visualization, optimization, robustness analysis, sensitivity analysis, reliability analysis and design space exploration [25].

Running a computer simulation of a complex system with multiple inputs and outputs can be a very time-consuming process. For example, Ford Motor Company reported on a crash simulation for a full passenger car that takes 36 to 160 hours to compute [25]. A computational expensive problem tackled in this thesis is the analysis and optimization of night cooling ventilation where one simulation run takes 6 to 96 hours [9] on an Intel® Core™ 2 quad core CPU Q9400@2.66 Ghz, 8 GB RAM (64 bit) desktop computer (utilizing all the cores). Because of these long computation times, analyzing such time-consuming simulations directly is still infeasible for engineers and scientists alike.

1.1.2. Input type

Most often input parameters can be assumed to be independent, i.e., changing one input parameter does not have an effect on the other input parameters. If a parameter is dependent on another (input) parameter analyzing the system may become more complex as it can actually be considered an output of the system. Therefore it is essential that the problem is formulated and parametrized correctly. Furthermore, input parameters can be either discrete or continuous. Discrete parameters denote, e.g., the presence or absence of a chemical component or the type of chemical component in a system.

1.1.2.1. Dimensionality

Perhaps the biggest hurdle in analyzing computational expensive simulation codes is the input dimension. The *curse of dimensionality* dictates that high-dimensional problems quickly become intractable, because data

spreads out exponentially with the number of dimensions. Therefore, analyzing high-dimensional data, even when there are millions of data points available, will only be possible to a certain degree.

The number of input parameters is a deciding factor in how to work with the simulation. While techniques may be developed that operate directly on the large number of input parameters. Many of those input parameters actually have little influence on the corresponding output of the system. In this case, variable and feature selection techniques [12] alleviate the computational complexity by selecting a much smaller subset of (relevant) input parameters to work with, i.e., the input parameters that have the largest effect on the output.

1.1.2.2. Uncertainty

Input parameters can also contain some uncertainty, e.g., the inputs are considered random variables with corresponding probability density functions. This stochastic variability is due to manufacturing deficiencies, imperfections, etc. in the system. In order to study the impact of stochastic variability on the response (performance), algorithms needs to be used that takes the uncertainty and statistical variation of the input parameters into account. These algorithms will make it possible to quantify the statistical importance of each input parameter and propagate this uncertainty to the response. A typical problem that contains uncertainty is to find an optimal design that is insensitive to small shifts in the input space due to manufacturing deficiencies. If the manufactured design deviates slightly from the identified optimal design the final produced product performance will still be close to optimal.

1.1.3. Output type

Similarly to input parameters, simulations can be broadly classified in the type of outputs they produce. Some simulation problems produce continuous outputs, while others output discrete values.

These latter type of simulation problems are called (statistical) classification problems where the goal is to map sets of inputs to the appropriate discrete output values, e.g., to 0 or 1 in binary classification problems. Standard examples are recognizing speech or handwritten text, faces of people in pictures or video, etc. This is a long-standing research subject originating from statistics, which was picked up by the machine learning community as supervised learning [24]. Many algorithms are available to tackle problems with discrete output values, such as decision trees, nearest neighbor algorithms, support vector machines [23], etc.

The other class of simulations are those with continuous outputs, known as regression analysis in statistics and machine learning [8, 18]. Because there is an unlimited number of possible output values, algorithms which

were designed for discrete output values do not directly apply to these problems.

1.1.3.1. Dimensionality

Usually the number of outputs of the simulation is of lesser importance. At least if they are independent of each other, then the same algorithms can be applied multiple times, once for each output. Should the outputs be correlated then it is much more efficient to recognize and exploit this correlation, though this is more of a challenge. Again, the presence of dependent output parameters primarily depends on how the problem is formulated. Sometimes this is unavoidable, for instance, many electromagnetic simulations output complex numbers where the real and imaginary components are often correlated.

1.1.3.2. Noise

Noise on the outputs of the simulation can make it considerable more difficult to analyze a system. There is an important different between deterministic noise and stochastic noise. The former, deterministic noise, is caused by inaccuracies in the simulation code. Performing simulations of the same set of input parameters multiple times results in the same output (with noise) values. On the other hand, stochastic noise is a result of random components, such as Monte Carlo methods, in the simulation code. This type of noise will produce different output values for the same set of input parameters over multiple calls to the simulation code. In stochastic simulation this noise needs to be taken into account in every stage of the analysis [26, 27]. For example, with stochastic noise it is useful and even necessary to apply classical statistical methods such as replication methods.

1.1.4. Black or white box

When nothing (or very little) is known up-front about the true behavior of the system, the simulation is called a black box. The only knowledge about the system is obtained from running simulations of the system, no assumptions about smoothness, monotonicity or other mathematical properties can be made. The system is solely studied using the input-output behavior (behavioral modeling) of the simulation. The advantage is that no explicit domain specific dependencies are introduced and, thus, any developed methods are readily portable across different problems and fields.

Alternatively, when the system is a gray or white box, prior knowledge about the inner workings is available. In general it is advisable to use this information to make more informed decisions and to use methods that are able to incorporate this information, enhancing the efficiency and expediting the analysis. This might even be required for certain problems,

e.g., in electromagnetics where it is often crucial to enforce certain properties such as monotonicity, time-invariance, etc.

1.2. Scope

The main driving force of this thesis is to make a study of time-consuming input-output based systems viable. To that end, simpler data-driven approximation models (in contrast to model-driven, e.g., model-order-reduction methods [11, 19]) are created to predict the system performance and develop a relationship between the inputs and outputs. These approximation models (also known as surrogate models, metamodels, emulators, replacement models, or response surface models) mimic the behavior of the simulation code while being computational cheap(er) to evaluate. This surrogate model is constructed by performing multiple simulations (called samples) at key points in the design space, see Figure 1.1.

While various surrogate modeling techniques exist that can cope with simulators with different properties we cannot possible provide a discussion for every possible system. Hence, this thesis is limited to problems and simulation models with the following properties: time-consuming, deterministic, regression problems (continuous outputs), spatial prediction (in contrast to temporal; time prediction), quasi noise-free and black box.

Examples of such simulation code are mechanical or electrical finite element simulations, computational fluid dynamic simulations, etc. The goal of this thesis is not to understand the underlying equation of a system, i.e., create a white box surrogate model from a black box simulation model [20]. The surrogate models in this thesis are used and analyzed instead of the time-consuming system.

1.2.1. Surrogate modeling

In (forward) surrogate modeling the engineer is interested in the output of the system, given an input. As stated before the principal reason to use surrogate models is that the simulator is too time consuming to run for a large number of simulations. One model evaluation may take many minutes, hours, days or even weeks [25]. A simpler approximation of the simulator is needed to make optimization, design space exploration, etc. feasible. Depending on the construction and use of approximation models several modeling flavors can be distinguished.

Global surrogate models capture the behavior of the system over the entire input space of interest. A global accurate surrogate model is an accurate approximation that can fully replace the original, computational expensive system in further analysis to gain insight. The cheap(er) surrogate model may be easily queried, optimized, visualized, and seamlessly integrated into CAD/CAE software packages. However, global surrogate models may

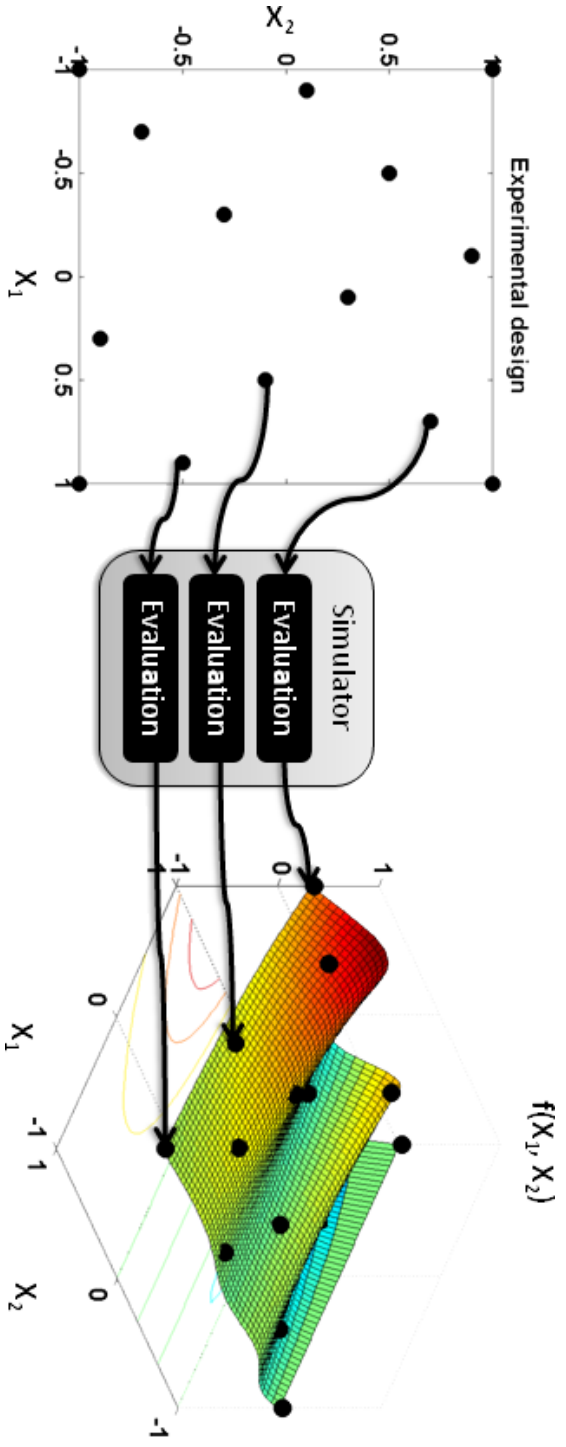


Figure 1.1: A set of data points is evaluated by a black box simulator, which outputs a response for every data point. An approximation model (surrogate model) is fit to the data points.

be created that are not necessarily a global accurate representation of the system. These types of surrogate models are usually just a means to an end to accomplish one specific task, e.g., optimization, sensitivity analysis, etc. The surrogate model can not fully replace the original system and is often discarded afterwards. Similarly, local surrogate models approximate a small part of the input space of interest and are mainly used in trust-region-like optimization schemes [2] which are useful to tackle large-scale problems.

The construction of surrogate models as efficiently as possible is an entire research domain in itself. In order to come to an acceptable model, numerous problems and design choices need to be overcome (what data collection strategy to use, which variables are relevant, how to integrate domain knowledge, etc.). Other aspects of surrogate modeling include choosing the right type of approximation model for the problem at hand, a tuning strategy for the surrogate model parameters (=hyperparameters), and a performance measure to assess the accuracy of the surrogate model [10]. There are a wide variety of surrogate model types available, and which one is most suitable depends largely on the system behavior that is to be modeled. Popular choices are polynomial and rational functions [6, 14], Kriging models [18, 15], neural networks [28] and radial basis functions (RBF) models [16].

In traditional Design of Experiments (DoE), the distribution of data points (experimental design) is chosen based only on information that is available before the first simulation, such as the existence of noise, the relevance of the input variables, the measurement precision, etc. This set of points is then fed to the simulator, which evaluates all the selected data points. Finally, a surrogate model is built using this data. This is essentially a one-shot approach, as all the data points are chosen at once and the surrogate modeling algorithm proceeds from there, without evaluating any additional samples later on.

In the deterministic black box setting, where there is no information available up front and statistical methods such as blocking and replication lose their relevance, the only sensible one-shot experimental designs are space-filling designs, which try to cover the design space as evenly as possible. The advantages of classical space-filling methods are that they can be easily implemented and provide a good (and guaranteed) coverage of the domain. Examples of popular space-filling design are fractional designs [22], Latin hypercubes [21] and orthogonal arrays [7]. On the other hand, in a one-shot design the number of points needs to be determined upfront which can easily lead to a sub-optimal choice: evaluating too few or too many points to achieve the desired accuracy. Sequential design, also known as adaptive sampling or active learning, improves on this approach by transforming the one-shot algorithm into an iterative process. Sequential design methods analyze data (samples) and surrogate models from previous iterations in order to select new samples in areas that are more difficult to approximate, resulting in a more efficient (and smaller) distribution of

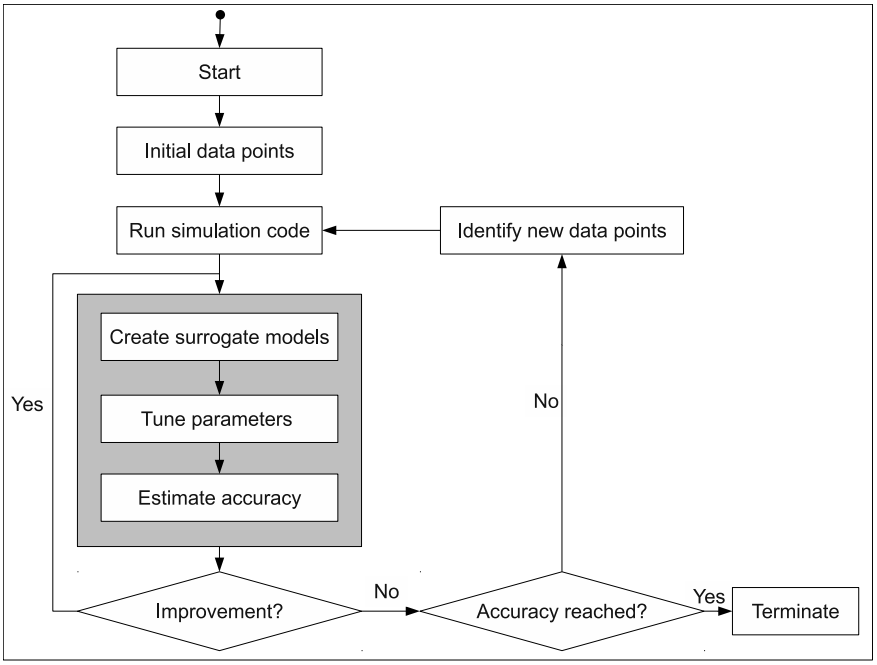


Figure 1.2.: Flow chart of the surrogate modeling process [10].

samples compared to traditional design of experiments. For example, the Lola-Voronoi algorithm selects points uniformly across the input space but also concentrates on regions which exhibit non-linear behavior [5].

The general work-flow of surrogate modeling is illustrated in Figure 1.2. First, an experimental design, e.g., from traditional DoE, is specified and evaluated. Subsequently, surrogate models are built to fit this data as well as possible, according to a set of measures (e.g., cross validation). The hyperparameters are estimated using an optimization algorithm. The accuracy of the set of surrogate models is improved until no further improvement can be obtained (or when another stopping criterion, such as a time limit, is met). If the stopping criteria are satisfied the process is halted and the final, best surrogate model is returned. On the other hand, when no stopping criterion is met, a sequential design strategy, also known as active learning or adaptive sampling, will select new data points to be evaluated and the surrogate models are updated with this new data.

Most often, surrogate models are used to solve so-called "*forward problems*". The practitioner is interested in the output or performance characteristics of the simulation system given the input (design) parameters. The surrogate models define the mapping between the input space (design space) and the output space (performance space). Examples of forward

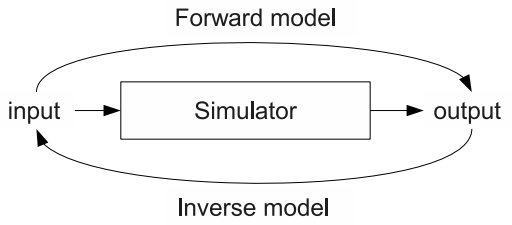


Figure 1.3.: Forward versus inverse modeling.

problems are found in validation and verification, sensitivity analysis, and optimization.

The operating context of this thesis is the integration of surrogate models into the optimization process, often denoted by Surrogate-Based Optimization (SBO) or Metamodel-Assisted Optimization (MAO). Note that the sequential design techniques employed in SBO methods also try to minimize the number of expensive samples but have a completely different goal, i.e., finding the global (or local) optimum instead of creating a global accurate surrogate model. The sequential design algorithm chooses points, based on previous surrogate models and data points, to guide the search towards the global (or local) optimal solution. SBO methods typically generate surrogate models on the fly that are only accurate in certain regions of the input space, e.g., around potentially optimal regions.

1.2.2. Inverse surrogate modeling

In inverse surrogate modeling the focus is on so-called "*reverse (inverse) problem*s", i.e., on exploring the input space. Ideally, a surrogate model could be created that maps the output parameters to the input parameters (as opposite to forward modeling) of the complex system over the entire output space, see Figure 1.3. However, many inverse problems are ill-posed. Considering Hadamard's definition of ill-posedness [13], the two outstanding problems hampering the creation of a full inverse surrogate model are non-uniqueness and instability. A good overview of the associated intricacies is presented by Barton in [3]. For all the above reasons, the inverse problem is often reduced to the task of finding one (or more) input parameter combinations for a certain output characteristic. Still, it is possible that,

1. no such input parameter combination exists,
2. more than one input parameter combination satisfies the given output characteristic(s).

A typical inverse problem is the estimation of some (physical) material or design parameter, e.g., the permittivity of a substrate [4] or the elasticity

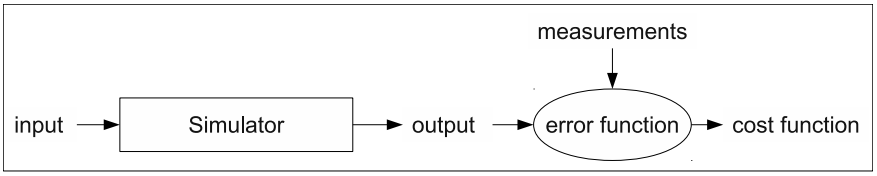


Figure 1.4.: The inverse problem is often solved by minimizing the error function between the simulation output and the measured data.

of rubber [1], given the desired output or system behavior. A popular solution is to convert the reverse problem to a (forward) optimization problem. Namely, a simulation model is constructed, parametrized by the properties or design parameters of interest. By minimizing the error between the parametrized simulation model and the measured data the input parameters (material properties) of the simulation model are obtained that correspond with the measurements or desired output, see Figure 1.4. On the other hand, sampling techniques can solve inverse problems directly leading to multiple solutions instead of just one, i.e., identify those regions in the input space that correspond to the desired output value(s).

1.3. Outline

This dissertation consists of six chapters and two appendices. Chapter 1 presents a general introduction to surrogate modeling and outlines the different chapters in this dissertation.

A significant contribution of this thesis is the flexible framework for surrogate modeling, i.e., the SUroggate MOdeling (SUMO) toolbox. The SUMO toolbox is implemented in Matlab and Java and widely used as a (forward) surrogate modeling tool, which resulted in various applications and research papers on surrogate modeling. The first part of Chapter 2 provides an more in-depth overview of global surrogate modeling and associated problems. In particular, the notion of building a surrogate model using multiple accuracy measures in a multiobjective way is explored. The second part of Chapter 2 describes the ooDACE toolbox which incorporates various types of the Kriging surrogate model for the first time in one framework using object oriented programming. The ooDACE toolbox is the foundation of several (optimization) algorithms presented in this thesis. This efficient Kriging implementation has been used on its own and as part of the SUMO toolbox by researchers over the world. One type of Kriging, i.e., blind Kriging, is thoroughly benchmarked on a very distinct set of real-life problems.

Chapter 3 gives an overview of the current state-of-the-art SBO algorithms and presents two applications optimized using a popular SBO algorithm,

i.e., the Efficient Global Optimization algorithm (EGO). The EGO algorithm, which forms the core of this thesis, is used to solve forward as well as inverse problems. The remaining part of Chapter 3 discusses the use of automatic surrogate model selection (using the Evolutionary Model Selection algorithm) for optimization. Traditionally, the EGO algorithm is used in conjunction with the Kriging surrogate model. Here, EGO is extended to automatically use the best surrogate model type such as support vector regression, radial basis functions, etc. and demonstrate its usefulness on an optimization problem from mechanical engineering.

Chapter 4 presents efficient methods to solve expensive multiobjective optimization problems with many objectives. The EGO algorithm is widely used for single-objective optimization though the concept is also very much suited for multiobjective optimization. Few work exists in literature as the generalization to multiobjective optimization of the EGO algorithm induces some computational difficulties. Novel algorithms are introduced that can be used to apply EGO to multiobjective optimization much more efficiently.

Chapter 5 deals with using surrogate models to solve inverse problems directly. Inverse problems are difficult to solve directly as they can be ill-conditioned. The most used approach is to reduce complex inverse problems to a more simple forward problem such as optimizing the error between the output of the simulation code and some target value or range. In this thesis a sampling strategy is developed, based on the Kriging surrogate model, that focuses on sampling the input regions densely that correspond to the desired target value or range.

Chapter 6 contains conclusions and future prospects. Lastly, appendix A and B discuss the SUMO and ooDACE toolbox in more detail, respectively.

1.4. Bibliography

[1] J. Aernouts, I. Couckuyt, K. Crombecq, and J.J.J. Dirckx. Elastic characterization of membranes with a complex shape using point indentation measurements and inverse modelling. *International Journal of Engineering Science*, 48:599–611, 2010.

[2] N. Alexandrov, J. E. Dennis, R. M. Lewis, and V. Torczon. A trust region framework for managing the use of approximation models in optimization. *Structural Optimization*, 15:16–23, October 1998.

[3] R.R. Barton. Issues in development of simultaneous forward-inverse metamodels. In *Proceedings of the Winter Simulation Conference*, pages 209–217, 2005.

[4] I. Couckuyt, F. Declercq, T. Dhaene, and H. Rogier. Surrogate-based infill optimization applied to electromagnetic problems. *Journal of*

RF and Microwave Computer-Aided Engineering: Advances in design optimization of microwave/rf circuits and systems, 20(5):492–501, 2010.

[5] K. Crombecq, D. Gorissen, D. Deschrijver, and T. Dhaene. A novel hybrid sequential design strategy for global surrogate modelling of computer experiments. *SIAM Journal of Scientific Computing*, 33(4):1948–1974, 2010.

[6] D. Deschrijver, T. Dhaene, and J. Broeckhove. Adaptive model based parameter estimation, based on sparse data and frequency derivatives. In *International Conference on Computational Science*, volume 3037. Springer, 2004.

[7] K-T. Fang, R. Li, and A. Sudjianto. *Design and Modeling for Computer Experiments*. Chapman & Hall/CRC, 2005.

[8] D. A. Freedman. *Statistical Models: Theory and Practice*. Cambridge Univer- sity Press, 2005.

[9] K. Goethals, I. Couckuyt, T. Dhaene, and A. Janssens. Sensitivity of night cooling performance to room/system design : surrogate models based on cfd. *Building and Environment*, 58:23–36, 2012.

[10] D. Gorissen, K. Crombecq, I. Couckuyt, P. Demeester, and T. Dhaene. A surrogate modeling and adaptive sampling toolbox for computer based design. *Journal of Machine Learning Research*, 11:2051–2055, 2010.

[11] S. Gugercin and A.C. Antoulas. A comparative study of 7 algorithms for model reduction. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pages 2367–2372vol.3, 12-15 Dec. 2000.

[12] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Machine Learning*, 3:1157–1182, 2003.

[13] J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. Technical Report 49–52, Princeton University Bulletin, 1902.

[14] W. Hendrickx and T. Dhaene. Sequential design and rational metamodelling. In M.E Kuhl, Steiger N. M., F. B. Armstrong, and J. A. Joines, editors, *Proceedings of the 2005 Winter Simulation Conference*, pages 290–298, Dec. 2005.

[15] J. P.C. Kleijnen. *DASE : Design and Analysis of Simulation Experiments*. Springer, May 2007.

[16] A. Lamecki, P. Kozakowski, and M. Mrozowski. CAD-model construction based on adaptive radial basis functions interpolation technique. 3:799–802, 2004. Microwaves, Radar and Wireless Communications, 2004. MIKON-2004.

[17] J. T. Oden. Revolutionizing engineering science through simulation. Technical report, National Science Foundation (NSF), Blue Ribbon Panel on Simulation-Based Engineering Science, 2006.

[18] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[19] M. Rewienski and J. White. Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations. *Linear algebra and its applications*, 415(2–3):426–454, 2006.

[20] S. Shan and G. Wang. Turning black-box functions into white functions. *Journal of Mechanical Design*, 133(3):1–10, March 2011.

[21] T. Simpson, D. Lin, and W. Chen. Sampling strategies for computer experiments: Design and analysis. *International Journal of Reliability and Application*, 2(3):209–240, 2002.

[22] T. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: Survey and recommendations. *Eng. Comput. (Lond.)*, 17(2):129–150, 2001.

[23] J. A. K. Suykens and J. Vandewalle. Multiclass least squares support vector machines. In *International Joint Conference on Neural Networks (IJCNN)*, pages 900–903, Jul 1999.

[24] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[25] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[26] D. Xiu. Fast numerical methods for stochastic computations: a review. *Communications in Computational Physics*, 5(2–4):242–272, 2009.

[27] D. Xiu. *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010.

[28] Q. J. Zhang and K. C. Gupta. *Neural Networks for RF and Microwave Design (Book + Neuromodeler Disk)*. Artech House, Inc., Norwood, MA, USA, 2000.

1.5. List of publications

1.5.1. International journals

[1] **I. Couckuyt**, J. Aernouts, D. Deschrijver, F. De Turck, and T. Dhaene. *Identification of quasi-optimal regions in the design space using surrogate modeling*. Engineering with Computers, January 2012.

[2] **I. Couckuyt**, F. Declercq, T. Dhaene, and H. Rogier. *Surrogate-Based Infill Optimization Applied to Electromagnetic Problems*. Journal of RF and Microwave Computer-Aided Engineering: Advances in design optimization of microwave/rf circuits and systems, 20(5):492–501, September 2010.

[3] **I. Couckuyt**, D. Deschrijver, and T. Dhaene. *Fast calculation of the Multiobjective Probability of Improvement and Expected Improvement criteria for Pareto Optimization*. Journal of Global Optimization, submitted.

[4] **I. Couckuyt** and T. Dhaene. *ooDACE Toolbox A Flexible Object-Oriented Kriging Implementation*. Journal of Machine Learning Research, accepted.

[5] **I. Couckuyt**, A. Forrester, D. Gorissen, F. De Turck, and T. Dhaene. *Blind kriging: Implementation and performance analysis*. Advances in Engineering Software, 49:1–13, 2012.

[6] **I. Couckuyt**, S. Koziel, and T. Dhaene. *Surrogate modeling of microwave structures using kriging, co-kriging, and space mapping*. International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 26(1):64–73, January/February 2012.

[7] J. Aernouts, **I. Couckuyt**, K. Crombecq, and J.J.J. Dirckx. *Elastic characterization of membranes with a complex shape using point indentation measurements and inverse modelling*. International Journal of Engineering Science, 48:599–611, 2010.

[8] G. Crevecoeur, A. Abdallh, **I. Couckuyt**, L. Depre, and T. Dhaene. *Two-level refined direct optimization scheme using intermediate surrogate models for electromagnetic optimization of a switched reluctance motor*. Engineering with Computers, 28(2):199–207, April 2012.

[9] J. Degroote, **I. Couckuyt**, J. Vierendeels, P. Segers, and T. Dhaene. *Inverse modelling of an aneurysm's stiffness using surrogate-based optimization and fluid-structure interaction simulations*. Structural and Multidisciplinary Optimization, 46(3):457–469, 2012.

[10] C. Gazda, **I. Couckuyt**, D. Vande Ginste, H. Rogier, .T Dhaene, Kristof Stijnen, and Hugo Pues. *Harmonic Balance Surrogate-Based Immunity*

Modeling of a Nonlinear Analog Circuit. IEEE Transactions on Electromagnetic Compatibility, accepted.

[11] C. Gazda, **I. Couckuyt**, H. Rogier, D. Vande Ginste, and T. Dhaene. *Constrained Multi-Objective Optimization of a Common-Mode Suppression Filter*. IEEE Transactions on Electromagnetic Compatibility, 54(3):704–707, 2012.

[12] K. Goethals, **I. Couckuyt**, T. Dhaene, and A. Janssens. *Sensitivity of night cooling performance to room/system design : surrogate models based on CFD*. Building and Environment, 58:23–36, 2012.

[13] D. Gorissen, **I. Couckuyt**, E. Laermans, and T. Dhaene. *Multiobjective global surrogate modeling,dealing with the 5-percent problem*. Engineering with Computers, 26(1):81–89, Jan. 2010.

[14] D. Gorissen, K. Crombecq, **I. Couckuyt**, P. Demeester, and T. Dhaene. *A Surrogate Modeling and Adaptive Sampling Toolbox for Computer Based Design*. Journal of Machine Learning Research, 11:2051–2055, 2010.

[15] S. Koziel, L. Leifsson, **I. Couckuyt**, and T. Dhaene. *Reliable reduced cost modeling and design optimization of microwave filters using co-kriging*. Int. J. Numerical Modelling: Electronic Devices and Fields, to appear.

[16] S. Koziel, L. Leifsson, **I. Couckuyt**, and T. Dhaene. *Robust variable-fidelity optimization of microwave filters using co-kriging and trust-regions*. Microwave and Optical Technology Letters, to appear.

[17] S. Koziel, S. Ogurtsov, **I. Couckuyt**, and T. Dhaene. *Variable-fidelity electromagnetic simulations and co-kriging for accurate modeling of antennas*. IEEE Trans. Antennas Prop., to appear.

[18] S. Koziel, S. Ogurtsov, **I. Couckuyt**, and T. Dhaene. *Cost-efficient EM-simulation-driven antenna design using co-kriging*. IET Microwaves, Antennas Prop., 6(14):1521–1528, 2012.

1.5.2. International conferences

[1] **I. Couckuyt**, J. Broeckhove, T. Dhaene, K. Crombecq, and D. Gorissen. *Analyzing and Optimizing Expensive Computer Simulations: The Surrogate Model Approach*. In International Summer School on Modelling and Optimization in Micro- and Nano- Electronics (Invited talk), 2008.

[2] **I. Couckuyt**, K. Crombecq, D. Gorissen, and T. Dhaene. *Automated Response Surface Model Generation with Sequential Design*. In First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering (CSC), Funchal, Portugal, 2009.

[3] **I. Couckuyt**, D. Deschrijver, and T. Dhaene. *Towards Efficient Multiobjective Optimization: Multiobjective statistical criterions*. In IEEE World Congress on Computational Intelligence, pages 1–7, 2012.

[4] **I. Couckuyt** and T. Dhaene. *Surrogate-based modeling of electrical systems*. In Advances in Modeling and Optimization of High Frequency Structures, International workshop, 2010.

[5] **I. Couckuyt**, D. Gorissen, F. DeTurck, and T. Dhaene. *Inverse Surrogate Modeling: Output Performance Space Sampling*. In 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, 2010.

[6] **I. Couckuyt**, D. Gorissen, H. Rouhani, E. Laermans, and T. Dhaene. *Evolutionary Regression Modeling with Active Learning: An Application to Rainfall Runoff Modeling*. In International Conference on Adaptive and Natural Computing Algorithms, volume LNCS 5495, pages 548–558, Sep. 2009.

[7] **I. Couckuyt**, D. Gorissen, F. De Turck, and T. Dhaene. *Automatic surrogate model type selection during the optimization of expensive black-box problems*. In Winter Simulation Conference, pages 4269–4279, 2011.

[8] **I. Couckuyt**, S. Koziel, and T. Dhaene. *Kriging, Co-Kriging and Space Mapping for Microwave Circuit Modeling*. In Proceedings of the 41st European Microwave Conference, 2011.

[9] K. Chemmangat, D. Deschrijver, **I. Couckuyt**, T. Dhaene, and L. Knockaert. *Fast Optimization of Microwave Filters using Surrogate-Based Optimization Methods*. In International Conference on Electromagnetics in Advanced Applications (ICEAA), pages 212–215, 2012.

[10] G. Crevecoeur, A. Abdallh, **I. Couckuyt**, D. Gorissen, L. Depre, and T. Dhaene. *Two-level refined direct method for electromagnetic optimization and inverse problems*. In Computation of Electromagnetic Fields, 17th conference, posters, 2009.

[11] K. Crombecq, **I. Couckuyt**, D. Gorissen, and T. Dhaene. *Space-filling Sequential Design Strategies for Adaptive Surrogate Modelling*. In Soft Computing Technology in Civil, Structural and Environmental Engineering (CSC 2009), 2009.

[12] K. Crombecq, **I. Couckuyt**, E. Laermans, and T. Dhaene. *A Novel Hybrid Active Learning Strategy for Nonlinear Regression*. In BeneLearn, 2009.

[13] F. Declercq, , **I. Couckuyt**, H. Rogier, and T. Dhaene. *Complex Permittivity Characterization of textile materials by means of Surrogate Modelling*. In IEEE International Symposium Antennas and Propagation, 2010.

[14] J. Degroote, **I. Couckuyt**, , J. Vierendeels, P. Segers, and T. Dhaene. *Inverse modelling of an aneurysm's elasticity using surrogate-based optimization of a three-dimensional fluidstructure interaction simulation*. In ECCOMAS Conference on CFD & Optimization, pages 1–16, May 2011.

[15] T. Dorné, F. Vanhee, T. Grenson, D. Pissoort, D. Deschrijver, **I. Couckuyt**, and T. Dhaene. *Optimized Sequential Sampling Algorithm for EMI Near-Field Scanning*. In EMC Europe, submitted.

[16] C. Gazda, **I. Couckuyt**, H. Rogier, D. Vande Ginste, and T. Dhaene. *Time Domain Analysis of a Common-Mode Suppression Filter Subjected to a Multi-Objective Optimization*. In EMC Europe, 2012.

[17] C. Gazda, D. Vande Ginste, H. Rogier, **I. Couckuyt**, T. Dhaene, K. Stijnen, and H. Pues. *An Immunity Modeling Technique to Predict the Influence of Continuous Wave and Amplitude Modulated Noise on Nonlinear Analog Circuits*. In EMC Europe, submitted.

[18] C. Gazda, D. Vande Ginste, H. Rogier, **I. Couckuyt**, T. Dhaene, K. Stijnen, and H. Pues. *Efficient Optimization of the Integrity Behavior of Analog Nonlinear Devices Using Surrogate Models*. In Signal and Power Integrity (SPI) workshop, 2013.

[19] K. Goethals, **I. Couckuyt**, T. Dhaene, and A. Janssens. *Sensitivity of night cooling performance to room/system design : surrogate models based on CFD*. In 5th International Building Physics Conference, pages 777–784, 2012.

[20] D. Gorissen, **I. Couckuyt**, K. Crombecq, and T. Dhaene. *Pareto-based multi-output model type selection*. In Proceedings of the 4th International Conference on Hybrid Artificial Intelligence (HAIS 2009), Salamanca, Spain, pages 442–449. Springer - Lecture Notes in Artificial Intelligence, Vol. LNCS 5572, 2009.

[21] D. Gorissen, **I. Couckuyt**, E. Laermans, and T. Dhaene. *Pareto-based multi-output metamodeling with active learning*. In Proceedings of the 11th International Conference on Engineering Applications of Neural Networks (EANN 2009), London, England, 2009.

[22] S. Koziel, **I. Couckuyt**, and T. Dhaene. *Reliable Low-Cost Co-Kriging Modeling of Microwave Devices*. In IEEE MTT-S I,t. Microwave Symp. Dig., 2012.

[23] S. Koziel, **I. Couckuyt**, and T. Dhaene. *Variable-fidelity optimization of microwave filters using co-kriging and trust regions*. In IEEE European Microwave Conference, 2012.

[24] S. Koziel, S. Ogurtsov, **I. Couckuyt**, and T. Dhaene. *Accurate Modeling of Antennas Using Variable-Fidelity EM Simulations and Co-Kriging*. In European Antenna and Propagation Conference, 2012.

[25] S. Koziel, S. Ogurtsov, **I. Couckuyt**, and T. Dhaene. *Efficient Simulation-Driven Design Optimization of Antennas Using Co-Kriging*. In International Symposium on Antennas and Propagation, 2012.

[26] H. M. Nguyen, **I. Couckuyt**, Y. Saeys, L. Knockaert, and T. Dhaene. *Avoiding overfitting in surrogate modeling: An alternative approach*. In Proceedings of the twentieth Belgian Dutch conference on machine learning, 2011.

[27] M. H. Nguyen, **I. Couckuyt**, L. Knockaert, and T. Dhaene. *An Alternative Approach to Avoid Overfitting for Surrogate Models*. In Winter Simulation Conference (WSC), 2011.

[28] H. Rouhani, D. Gorissen, **I. Couckuyt**, and J. Feyen. *Automatic Calibration of Semi-Distributed Conceptual Rainfall-Runoff Model Using MOSCEM Algorithm*. In Proceedings of the 8th International Congress on Civil Engineering, Shiraz, Iran, 2009.

[29] B. Van der Streeck, F. Vanhee, B. Boesman, D. Pissoort, D. Deschrijver, **I. Couckuyt**, and T. Dhaene. *Practical implementation of a sequential sampling algorithm for EMI near-field scanning*. In International Symposium on Electromagnetic Compatibility, 2012.

[30] S. L. Teske, **I. Couckuyt**, T. J. Schildhauer, S. M.A. Biollaz, and F. Maréchal. *Integrating Rate Based Models into Multi-Objective Optimisation of Process Designs using Surrogate Models*. In International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems, 2013.

1.5.3. Book chapters

[1] D. Gorissen, K. Crombecq, **I. Couckuyt**, and T. Dhaene. *Foundations of Computational Intelligence, Volume 1: Learning and Approximation: Theoretical Foundations and Applications*, volume 201, chapter Automatic Approximation of Expensive Functions with Active Learning, pages 35–62. Springer Verlag, Series Studies in Computational Intelligence, 2009.

1.5.4. Technical reports

[1] D. Gorissen, **I. Couckuyt**, and T. Dhaene. *A Linear Reference Model (LRM) Metric for Model Selection and Sequential Design*. Technical report, University of Antwerp, 2009.

[2] D. Gorissen, **I. Couckuyt**, and T. Dhaene. *Multiobjective global surrogate modeling*. Technical Report TR-08-08, University of Antwerp, Middelheimlaan 1, 2020 Antwerp, Belgium, 2008.

[3] D. Gorissen, K. Crombecq, **I. Couckuyt**, and T. Dhaene. *Automatic Approximation of Expensive Functions with Active Learning*. Technical Report TR-10-08, University of Antwerp, Middelheimlaan 1, 2020 Antwerp, Belgium, 2008.

2 Forward surrogate modeling

Insanity: doing the same thing over and over again and expecting different results.
— Albert Einstein

2.1. Multiobjective global surrogate modeling, dealing with the 5-percent problem

D. Gorissen, I. Couckuyt, E. Laermans, T. Dhaene

Published in Engineering with Computers,

vol. 26, no. 1, pp. 81-89, 2010.

Abstract

When dealing with computationally expensive simulation codes or process measurement data, surrogate modeling methods are firmly established as facilitators for design space exploration, sensitivity analysis, visualization, prototyping and optimization. Typically the model parameter (=hyper-parameter) optimization problem as part of global surrogate modeling is formulated in a single objective way. Models are generated according to a single objective (accuracy). However, this requires an engineer to determine

a single accuracy target and measure upfront, which is hard to do if the behavior of the response is unknown. Likewise, the different outputs of a multi-output system are typically modeled separately by independent models. Again, a multiobjective approach would benefit the domain expert by giving information about output correlation and enabling automatic model type selection for each output dynamically. With this paper the authors attempt to increase awareness of the subtleties involved and discuss a number of solutions and applications. In particular we present a multiobjective framework for automatic global surrogate model generation to help tackle both problems and that is applicable in both the static and sequential design (adaptive sampling) case.

2.1.1. Introduction

Regardless of the rapid advances in High Performance Computing and multi-core architectures, it is rarely feasible to explore a design space using high fidelity computer simulations [93]. As a result, data based surrogate models (otherwise known as metamodels, emulators, or response surface models) have become a standard technique to reduce this computational burden and enable routine tasks such as visualization, design space exploration, prototyping, sensitivity analysis, and of course, optimization [98, 87].

It is important to first comment on the difference between local and global surrogate models since motivation and philosophy are distinct. Local surrogate modeling involves building small, relatively low fidelity surrogates for use in optimization. Local surrogates are used as rough approximators of the (costly) optimization surface and guide the optimization algorithm towards good extrema while minimizing the number of simulations [6]. Once the optimum is found, the surrogate is discarded. Many advanced methods for constructing and managing these local surrogates have been designed, including trust region methods [100, 1], various ensemble techniques [33], space mapping methods [20], and hierarchical surrogates [104]. In general the theory is referred to as Surrogate-Based Optimization (SBO) or Metamodel Assisted Optimization (MAO). A good overview reference is given by [19], [77], and the work by Y. S. Ong [73].

In contrast, with global surrogate modeling the surrogate model itself is usually the goal. The objective is to construct a high fidelity approximation model that is as accurate as possible over the complete design space of interest using as few simulation points as possible. Once constructed, the global surrogate model (also referred to as a replacement metamodel) is reused in other stages of the computational science and engineering pipeline. For example as a cheap accurate and scalable replacement model in standard design software packages (e.g., [11]). Thus optimization is usually not the main goal (though it still can be), but rather a useful post-processing step.

Of course the dichotomy is not strict; ideas and approaches between the

two types can, and should, be exchanged, allowing for different hybrids to emerge that borrow ideas from both types. A good example in this respect is the popular Efficient Global Optimization (EGO) approach first described by Jones et al. in [51] and elaborated by many others (e.g., [83]).

The current paper attempts to increase the value and utility of global surrogate methods for practitioners by exploring a multiobjective approach to surrogate model generation. This gives an engineer or domain expert more flexibility in specifying a priori constraints on the surrogate model generation process (cfr *"The 5 percent problem"* in Section 2.1.3). In addition, a multiobjective approach allows multi-output problems to be modeled directly, giving more information than modeling each output independently. At the same time we do not assume a fixed sample distribution but select and perform simulations iteratively (adaptive sampling) as would be the case in any real application.

2.1.2. Global surrogate modeling

We stress again that the context of this work is to efficiently generate accurate *global* surrogates (valid over the complete design space) using a minimal number of computationally expensive simulations. Optimization of the simulation output is not the main goal, rather we are concerned with optimization of the model parameters (hyperparameter optimization).

Global surrogate models are particularly useful for design space exploration, sensitivity analysis, prototyping, visualization, and *what-if* analysis. They are also widely used to build model libraries for use in controllers or engineering design software packages. In addition, they can cope with varying boundary conditions. This enables them to be chained together in a model cascade in order to approximate large scale systems [5]. A classic example is the full-wave simulation of an electronic circuit board. Electromagnetic modeling of the whole board in one run is almost intractable. Instead the board is modeled as a collection of small, compact, accurate surrogates that represent the different functional components (capacitors, transmission lines, resistors, etc.) on the board. Finally, if optimization is the goal, one could argue that a global model is less useful since significant time savings could be achieved if more effort were directed at finding the optimum rather than modeling regions of poor designs. However, this is the logic of purely local models, but they forgo any wider exploration of radical designs [30]. Some examples of global modeling approaches can be found in [71, 32, 7, 88].

The mathematical formulation of the problem is as follows: approximate an unknown multivariate function $f : \Omega \mapsto \mathbb{C}^n$, defined on some domain $\Omega \subset \mathbb{R}^d$, whose function values $f(X) = \{f(x_1), ..., f(x_k)\} \subset \mathbb{C}^n$ are known at a fixed set of pairwise distinct sample points $X = \{x_1, ..., x_k\} \subset \Omega$. Constructing an approximation requires finding a suitable function s from an approximation space S such that $s : \mathbb{R}^d \mapsto \mathbb{C}^n \in S$ and s closely resembles f

as measured by some criterion ξ. The task is then to find the best approximation $s^* \in S$ such that s^* satisfies $\min_{s \in S} \xi = s^*$. This minimization is an optimization problem over the model parameters, commonly referred to as the hyperparameter optimization problem. This may be solved manually, through trial and error, or using readily available optimization algorithms. Additional assumptions are that f is expensive to compute. Thus the number of function evaluations $f(X)$ needs to be minimized and data points must be selected iteratively, at points where the information gain will be the greatest [95]. Mathematically this means defining a sampling function

$$\phi(X_{i-1}) = X_i \, , \, i = 1, .., N \tag{2.1}$$

that constructs a data hierarchy

$$X_0 \subset X_1 \subset X_2 \subset ... \subset X_N \subset X \tag{2.2}$$

of nested subsets of $X = \{x_1, ..., x_k\}$, where N is the number of levels. X_0 is referred to as the *initial experimental design* and is constructed using one of the many algorithms available from the theory of Design and Analysis of Computer Experiments (DACE) (see the work by Kleijnen et al. [55]). Once the initial design X_0 is available it can be used to seed the sampling function ϕ. An important requirement of ϕ is to minimize the number of sample points $|X_i| - |X_{i-1}|$ selected each iteration (f is expensive to compute), yet maximize the information gain of each successive data level. This process is referred to as adaptive sampling [14], but is also known as active learning [16], reflective exploration [11], Optimal Experimental Design [79] and sequential design [53]. The advantage of adaptive sampling is that the number of required data points need not be specified up front, avoiding potential over- or undersampling. At the same time, by intelligently choosing the location of each data point the accuracy of the surrogate may be maintained. An important consequence of the adaptive sampling procedure is that the task of finding the best approximation s^* becomes a dynamic problem instead of a static one. Since the optimal model parameters will change as the amount and distribution of data points changes.

2.1.3. *"The 5 percent problem"*

The basic algorithm for generating a global surrogate model through adaptive sampling is as follows: a small number of simulations are performed according to some Design of Experiment plan. Given these sample points the space of candidate models S is searched for the besting fitting model s^* according to ξ. If s^* is acceptable (i.e., the model meets the target requirement set out by the user) the algorithm terminates. Else the sampling function ϕ is used to generate a new set of sampling points (adaptive sampling) and the model search is resumed. This whole process continues until

the user-defined accuracy has been reached or the computational budget is exhausted.

A crucial aspect of this algorithm is identifying a suitable criterion ξ, where ξ constitutes three parts:

$$\xi = (\Lambda, \varepsilon, \tau) \tag{2.3}$$

with Λ the generalization estimator, ε the error (or loss) function used, and τ the target value required by the user. This means that the global surrogate model generation problem (namely finding s^*) for a given set of data $D = (X_i, f(X_i))$ can be formally defined as

$$s^* = \arg\min_{t \in T} \arg\min_{\theta \in \Theta} \Lambda(\varepsilon, s_{t,\theta}, D) \tag{2.4}$$

such that

$$\Lambda(\varepsilon, s^*_{t,\theta}, D) \leqslant \tau \tag{2.5}$$

where $s_{t,\theta}$ is the parametrization θ (from a parameter space Θ) of s and $s_{t,\theta}$ is of model type t (from a set of model types T).

The outer minimization over $t \in T$ is the task of selecting a suitable approximation model type, i.e., a rational function, a neural network, a spline, etc. This is the model type selection problem. In practice, one typically considers only a single $t \in T$, though others may be included for comparison. Then given a particular approximation type t, the task is to find the hyperparameter assignment θ that minimizes the generalization estimator Λ (e.g., determine the optimal order of a polynomial model). This is the hyperparameter optimization problem, though generally both minimizations are simply referred to as the model selection problem.

Many implementations of Λ have been described: the hold-out, bootstrap, cross validation, jack-knife, Akaike Information Criterion (AIC), etc. In the simple case where Λ is just the in-sample error, the problem simplifies to

$$s^* = \min_{t \in T} \min_{\theta \in \Theta} \varepsilon(s_{t,\theta}(X_i), f(X_i)) \tag{2.6}$$

such that

$$\varepsilon(s^*_{t,\theta^*}(X_i), f(X_i)) \leqslant \tau \tag{2.7}$$

The crucial problem is identifying suitable implementations for Λ, ε and a target value for τ. The three are closely linked but only the Λ-selection problem has been extensively studied theoretically [2, 103] and empirically [89, 67]. The selection of ε and τ is less appreciated and often overlooked, but equally important [60, 26]. Selecting an error function ε and required target accuracy τ is difficult since it requires knowledge of the structure of the response and a full understanding of what the generalization estimator

Λ actually measures. Failure to do so leads to misinterpretation, inappropriate application, ultimately resulting in a trial and error model generating procedure.

This brings us to, what we have termed, *"The 5 percent problem"*. The phrase stems from an application where an engineer needed a replacement metamodel. When asked what model accuracy was required the answer was simply 5 percent. While this may seem like a straightforward, objective requirement, enforcing it in practice turned out to be difficult. The reason is twofold and is detailed below.

2.1.3.1. Choice of error function

First, there is the choice of the error function ε. Roughly speaking there are two categories of error functions: absolute and relative.

Absolute errors Absolute errors (e.g., Average Absolute Error (AAE), Mean Squared Error (MSE), etc.) are often undesirable in an application context since they are not unit-free and depend on the specific prediction value of the response. On the other hand they are very popular in machine learning settings but not always rightly used. A good example is the Root Mean Square Error (RMSE), by far the most popular error function:

$$RMSE(y, \tilde{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2} \tag{2.8}$$

Where $y_i, \tilde{y}_i$ are the real and predicted response values respectively. The main advantage of the RMSE is that it is the best finite-sample approximation of the standard error $\sqrt{E[y - \tilde{y}]}$ and standard deviation (in the case of an unbiased model) [60]. However, its use is not recommended since it penalizes large errors too severely while virtually ignoring small errors. Such an error function is called pessimistic. Also it is unintuitive to interpret. The RMSE is often interpreted as the true arithmetic average euclidean distance between the prediction $\tilde{y}$ and the true value y. This is however not the case, the RMSE is really one $\sqrt{n}$-th of this value and thus has no simple geometrical interpretation whatsoever. A better solution would be to use the Average Euclidean Error (AEE) as proposed by [60]

$$AEE(y, \tilde{y}) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(y_i - \tilde{y}_i)^2} \tag{2.9}$$

However, while the AEE enjoys many desirable properties, it still suffers from outliers (i.e., it is also pessimistic, though less than the RMSE). In cases where this is a problem alternative functions like the Geometric Average Error (GAE) and the Harmonic Average Error (HAE) [60] can be more useful.

$$GAE(y,\tilde{y}) = \left(\prod_{i=1}^{n}\sqrt{(y_i-\tilde{y}_i)^2}\right)^{\frac{1}{n}} \tag{2.10}$$

$$\begin{aligned} HAE(y,\tilde{y}) &= \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\sqrt{(y_i-\tilde{y}_i)^2}}\right)^{-1} \\ &= \frac{n}{\frac{1}{\sqrt{(y_1-\tilde{y}_1)^2}}+\ldots+\frac{1}{\sqrt{(y_n-\tilde{y}_n)^2}}} \end{aligned} \tag{2.11}$$

In contrast to the RMSE and AEE, the HAE is an optimistic error function since it is dominated by the small error terms. The HAE can be appropriate if the error fluctuates greatly over different runs. This property may be useful in the context of k-fold cross validation with relatively few samples. The GAE, on the other hand, is a balanced error function that suffers much less from extremes (large or small). The GAE, however, has as a disadvantage that if the error is zero in a single point, the overall error is also zero. This is of course may not be desirable.

Many more absolute error variants exist and we do not intend to give an exhaustive overview. Rather we wish to illustrate that each function brings its own tradeoffs and interpretation depending on how the absolute differences $|y_i - \tilde{y}_i|$ are aggregated. In general though, absolute error criteria are not ideally suited for performance estimation of an approximation model due to their context dependence (i.e., τ is hard to specify up front and depends on the units used).

Relative errors Thus engineers typically prefer relative or percentage errors, e.g., 5%. A figure of 5% implies some kind of global averaged relative error, but there are different ways to calculate relative errors (depending on what reference and aggregation function is used): Average Relative Error (ARE), Maximum Relative Error (MRE), Relative Squared Error (RSE), Root Relative Square Error (RRSE), Relative Absolute Error I (RAEI), Relative Absolute Error II (RAEII), Root Mean Square Relative Error (RMSRE), etc. [32, 60].

A natural solution is to take the most intuitive error function, the ARE:

$$ARE(y,\tilde{y}) = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i-\tilde{y}_i|}{|y_i|} \tag{2.12}$$

By taking the true value as a reference the ARE results in an intuitively understandable number. Multiplied by 100 it results in a natural percentage. However, taking the geometric or harmonic mean (resulting in the Geometric average Relative Error (GRE) and Harmonic average Relative Error (HRE) respectively) instead of the simple average can also be interpreted

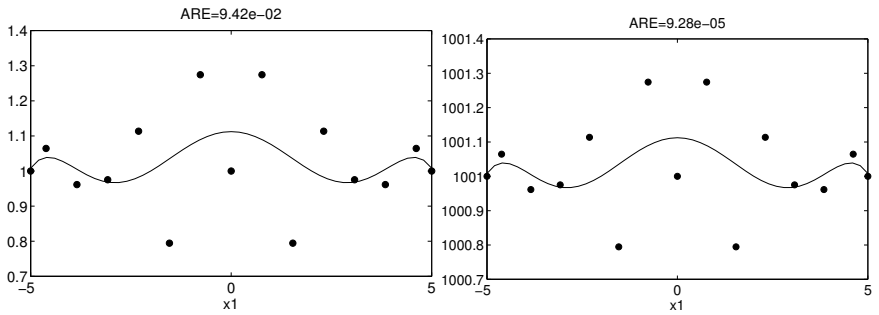


Figure 2.1.: Influence of shifting the response on the ARE.

as a global percentage error. But since ARE, GRE, and HRE treat different types of errors differently (e.g., ARE is more sensitive to large errors than GRE) care should be taken when interpreting a figure like 5%. In addition, the "%" suffix is sometimes also used when using, for example, RRSE (see below). This, however, should be avoided since it is confusing.

The problem with the ARE is that care must be taken in its interpretation when the true function values y_i are small or, in the extreme but not unlikely case, zero. Since then the error tends to infinity, giving a biased result. What is sometimes done to combat this is add one $(+1)$ to the denominator to prevent division by zero. While this solves the numerical issue, the resulting error is an absolute-relative hybrid and becomes impossible to interpret. A different solution is to scale or translate the response to eliminate small absolute values (e.g., as proposed in [42]). However, the best scale factor is not always obvious and shifting the response can introduce its own problems. For example, Figure 2.1 illustrates how a simple shifting of the response (+1000) can drastically improve the ARE value (3 orders of magnitude) for exactly the same model parameters (error measured in the samples).

Additionally, there is the well known issue of averaging the errors (relative or absolute). This means that a model with a low average error can still have areas where the prediction is very poor (i.e., the mean is not robust). Figure 2.2 shows an example using relative errors. The data in the figure are the result of a NIST study involving semiconductor electron mobility. The response variable is a measure of electron mobility, and the predictor variable is the natural log of the density. The fit is a rational function with a pole. Thus, since an engineer usually requires strict bounds on the maximum error it seems better to minimize the maximum error instead of the average (note that ARE $\leqslant$ MRE).

However, in the relative case, using a maximum aggregation function has its own counter-intuitive properties. For example, Figure 2.3 illustrates how the zero function has a lower MRE than a model which overshoots

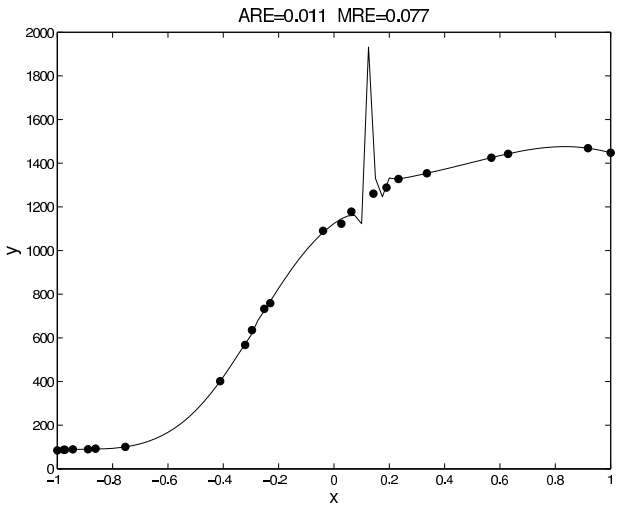


Figure 2.2.: MRE vs ARE.

the data, but else seems like a reasonable fit[1]. This property is particularly problematic if the model parameter space is searched automatically (hyper-parameter optimization). In this case the optimization algorithm is easily deceived into generating flat models.

One would be tempted to resort to using the Maximum Absolute Error (MAE) instead. Since while it may be difficult to give a priori average error targets, giving maximum absolute error bounds is often easier since it can be related more directly to the application. However, the MAE is not a satisfactory solution either. First of all, like any absolute error, it requires knowledge of the full range of the response. Also, it is not relative, meaning a deviation of 5 on a response value 1000 is considered worse than a deviation of 3 on a value of 0.5. Furthermore, enforcing a MAE is equivalent to restricting all fitted response values $\tilde{y}$ to lie inside the tube defined by $[y - MAE, y + MAE]$. This requirement can be too strict if the response contains regions that are very hard to fit (e.g., discontinuities), information that is not always available.

Another approach then, is to use the RRSE function, related to the popular R^2 criterion. In this case the deviation from the mean is used as the reference value.

$$RRSE(y, \tilde{y}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.13}$$

The RRSE is intuitively attractive since it measures how much better a fit

[1] In this case the samples are noisy but the same phenomenon can occur with noise free data and a validation set.

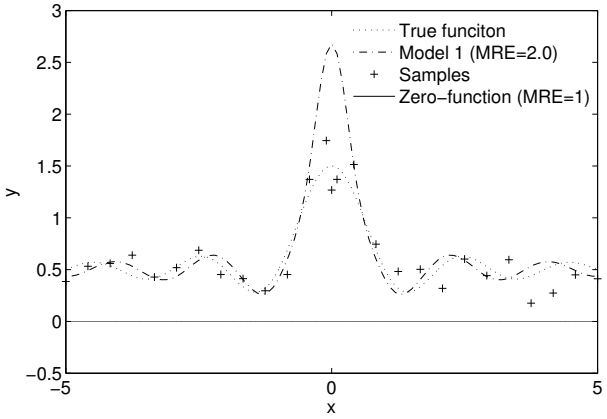


Figure 2.3.: Comparison of the MRE over different models.

is over the most simple model possible: the mean. Also it does not become problematic for small absolute values of y_i. Unfortunately, the problem with the RRSE is that it gives a pessimistic estimate of the error if the response that needs to be fitted is very smooth (i.e., the mean is already quite a good fit). Thus an understanding of the structure of the response is needed to properly interpret the RRSE value. The RRSE is also less intuitive for an engineer since it measures the improvement over the average model rather than the quality of fit directly (making a good choice of τ harder).

An improved function that is less sensitive to large errors and has some other attractive properties is given in [60], the Bayesian Estimation Error Quotient (BEEQ):

$$BEEQ(y, \tilde{y}) = \left(\prod_{i=1}^{n} \frac{\sum_{i=1}^{n} |y_i - \tilde{y}_i|}{\sum_{i=1}^{n} |y_i - \bar{y}|} \right)^{\frac{1}{n}} \tag{2.14}$$

However, like the GAE it will predict an error of zero overall if just a single point has an error of 0.

One could continue discussing different error functions (e.g., those based on the median or mode) but it should be clear now that each error function has its own characteristics and that relative errors are not always as context free as one might assume at first. While the examples given here are quite simple, they are illustrative of the greater complexities that arise when combining an error function with a model selection metric. Also note that these subtleties are less a problem in classification (where most research on model selection is conducted). The concept of a good classifier is typically much more intuitive to grasp and define by a domain expert than in the case of regression.

Remark that the error function also influences the choice of sampling

strategy. For example if the error measure dictates that it is important that the optima of the model are captured accurately, one should make sure the sampling strategy employed will sample at those locations. Actually it turns out that in most cases a sampling algorithm can be formulated as a model selection criterion and vice versa.

2.1.3.2. Choice of model selection metric

Assuming the choice of error function (and target value) can be decided upon there is still the problem of selecting a measure for estimating the generalization capabilities of a model (cross validation, bootstrap, validation set, jack-knife, etc.). This is the well known problem of model selection and has been discussed at length elsewhere [8, 2, 70, 89, 34]. A good high level introduction is given in [105]. The point this paper attempts to make is that it is far from obvious which method to select that, when minimized, produces a model that an engineer is satisfied with. Simply using the in-sample error is useless since it does not account for over-fitting and is meaningless when used with interpolating methods (e.g., Kriging). Measures like AIC and its variants (BIC, CIC, NIC, ...) and the methods from statistical learning theory (Vapnik-Chervonenkis (VC) Dimension, etc.) are more advanced in that they take the complexity of the model into account and have solid foundations in information theory. Unfortunately, an AIC value can only be interpreted relative to another value and has no meaning on its own. These type of measures also mean very little to a domain expert.

A validation (hold-out) set is a better solution but it means there is less data available for training. Also, the hold-out error can give extremely biased results (thus deceiving the hyperparameter optimization) if chosen poorly or if only a few points are available. For example, Figure 2.4 gives a simple example were minimizing the validation error can lead to a sub-optimal model. This is of course an extreme example but similar problems are often encountered with real data.

The cross validation error (and its extreme version, the Leave-One-Out error), is a popular compromise, but it too depends on the data distribution [17, 70], can give misleading results [61], and is expensive to compute (the bootstrap even more so). Also there is the question on how to select the folds (randomly, evenly spread, etc.). Additionally one could argue the different cross validation variants should be interpreted as measuring sensitivity to loss of information rather than approximation accuracy. Finally there is the added complication of noise in the data and/or in the generalization estimator (e.g., k-fold cross validation). Since we only consider deterministic computer experiments noisy data is usually not an issue[2]. However, when dealing with measured data or stochastic simulations this adds an extra layer of complexity.

[2] In some cases discretization and convergence noise may be present, the magnitude depending on the application.

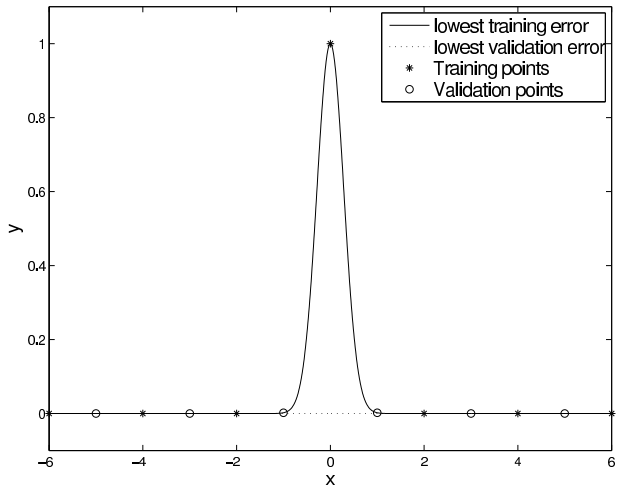


Figure 2.4.: A misleading validation set.

Yet a different approach is to employ Bayesian statistics (see the work by O'Hagan et al. [72]). Through Bayesian inference one can exactly quantify the uncertainty or confidence one has in a particular model. This is usually very useful from an application standpoint but is only possible with specific model types.

The only true, unbiased test for model quality would be to assess the model on a very dense, independent test set or analytical solution. However, for any real problem this is not a feasible option since data is too expensive and an analytical solution is not available.

2.1.3.3. The need for handling multiple criteria

In sum, as it should be clear now, it is hard to agree upfront on a single requirement that the final replacement metamodel must respect. The fundamental reason is that an approximation task inherently involves multiple, conflicting, criteria. [60] summarizes this particularly succinctly:

> *It is an illusion that performance evaluation can be done completely fairly and impartially. This is partly because simple metrics cannot capture a complete picture of the performance of an estimation algorithm and those that are more complete [...] are more complex and subject to subjective interpretations. Also, use of any metric in performance evaluation implicitly favors the estimator that tries to optimize this same metric.*

Thus what usually happens in practice is the following: (1) a best effort is made to identify a suitable model selection metric, error function and

targets; (2) simulations are performed, the model is generated and delivered to the engineer together with some statistical test results (e.g., different error metrics); and (3) the engineer visually inspects and explores the model and decides whether it is satisfactory. If not the process must be repeated.

While the final evaluation stage by a domain expert should always be performed, it would be advantageous if the different desired criteria could be enforced from the start. This can be done in three main ways:

1. the different criteria (objectives) are combined into a single, global criterion which is then used to drive the model generation

2. the different objectives are enforced sequentially in a multi-level process

3. the different objectives are enforced simultaneously through a multi-objective approach

The first option is the easiest and allows existing algorithms to be re-used as is. An example of such scalarization is the geometric mean approach used by Goel et al. in [33]. However the problem remains of choosing an appropriate combination function (and its interpretation) and requiring an understanding of the ranges and nuances of the different member functions. Thus the problem is simply moved to a higher level.

The second option is a sequential or milestone approach. Multiple criteria are supported by specifying different hierarchical levels $L_1, ..., L_l$ that must be reached in succession. For example, first the hyperparameter optimization process must produce a model that satisfies L_1 (e.g., a ARE of 5%). Once this target is reached, and only then, is the following level L_2 checked (e.g., a MRE of 10%). Thus, by sequentially working towards subsequent milestones, multiple criteria can be incorporated. The potential problem of this approach is that dependencies and tradeoffs between criteria can cause a deadlock (e.g., reaching one level means violating another). A different way to interpret this is as a constrained optimization problem in the hyperparameter space, each level adds a constraint. Care must be taken that there is at least one feasible region. Also the task for the optimizer (over the model parameters) becomes considerably more difficult since the optimization landscape may change suddenly and drastically when a change in level takes place.

The third option is to tackle the problem directly as a dynamic multiobjective optimization problem in the hyperparameter space (recall that due to the incremental sampling the optimization surface is dynamic and not static). Each criterion becomes an objective and standard ranking methods are used to identify the Pareto-optimal set. The disadvantage here is that there is no longer the luxury of having a single, unambiguous best solution. However, since we noted above that such a linear ranking makes no sense this should come as no surprise. The advantage is that the problem can be

tackled directly using standard algorithms. From the final Pareto set the user is then able to extract knowledge about the problem and make a better decision when choosing the final solution. In addition, the final Pareto front enables the generation of diverse ensembles, where the ensemble members consist of the (partial) Pareto-optimal set (see references in [33, 81, 50]). In this way all the information in the front can be used. An additional advantage of using ensembles is that it allows the calculation of the prediction uncertainty which is very useful for an application.

Finally, one may imagine different hybrid combinations of the three methods mentioned above. For example, the multiobjective approach where the number of objectives varies dynamically. For example, when only little data is available it makes no sense to enforce application specific criteria, or force the model response into particular bounds. That makes more sense when sufficient data is available and the model uncertainty has been reduced. Other combinations are possible but this is a topic that has seen little research and that goes beyond the scope of this chapter. Rather we shall concentrate on the multiobjective approach.

2.1.4. Modeling multiple outputs

The previous section described how a multiobjective approach to global surrogate modeling can help solve *the 5 percent problem*. A second use case is when dealing with multi-output systems. It is not uncommon that a simulation engine has multiple outputs that all need to be modeled [10]. For example, the combustion problem described in [45] has both a chemical and temperature source term that needs to be modeled. Also many Finite Element packages generate multiple performance values simultaneously.

The direct approach is to model each output independently with separate models (possibly sharing the same data). This, however, leaves no room for trade-offs nor gives any information about the correlation between different outputs. Instead of performing two modeling runs (doing a separate hyperparameter optimization for each output) both outputs can be modeled simultaneously if models with multiple outputs are used in conjunction with a multiobjective optimization routine. The resulting Pareto front then gives information about the accuracy trade-off between the outputs in hyperparameter space and allows the practitioner to choose the model most suited to the particular context. More arguments, of essentially the same discussion, are given in [66].

Again, multi-output Pareto based modeling enables the generation of diverse ensembles. This is a popular approach in rainfall runoff modeling and model calibration in hydrology [90, 25]. Models are generated for different flow components and/or derivative measures and these are then combined into a weighted ensemble or fuzzy committee. A Pareto based approach to multi-output modeling also allows integration with the automatic surrogate model type selection algorithm described in [40]. This enables automatic

selection of the best model type (Artificial Neural Network (ANN), Kriging, Support Vector Machine (SVM), ...) for each output without having to resort to multiple runs [37, 38].

2.1.5. Related work

There is a vast body of research available on single objective hyperparameter optimization strategies and model selection criteria for different model types: [9, 59, 94, 89, 18, 4, 74] and the extensive work by Yao et al. [101, 102]. Many authors have noticed the problems with single objective hyperparameter optimization but it is only very recently that multiobjective versions of classical machine learning methods have been presented [69, 91, 31, 48]. An extensive and excellent overview of the work in this area is given by Jin et al. in [50] and the book (edited by Jin) [47]. By far the majority of the cited work uses multiobjective techniques to improve the training of learning methods. Typically an accuracy criterion (such as the validation error) is used together with some regularization parameter or model complexity measure (e.g., the number of support vectors in SVMs) in order to produce more parsimonious models [26]. Other criteria used include: sensitivity, specificity, interpretability, and number of input features [91, 50].

It seems less work has been done on high level objectives (with error functions and generalization estimators in particular) that do not depend on a particular machine learning method. [27] optimize an accuracy metric (the RMSE) together with an application specific *Return* metric useful for stock market forecasting. An example of the use of multiple error measures (and incidentally one of the first formulations of multiobjective learning) is [63] who minimized the L_2-norm, the L_∞-norm and a complexity measure. Unfortunately, a single-objective GA was employed to perform the optimization, resulting in only a single solution. [28] also give an example with two error functions, the Euclidean and robust error which they use to fit a noisy sinusoid with an ANN.

Few references are available that explicitly deal with the trade-offs between different error functions for surrogate modeling. [26] agree that determining the error function is key but do not consider the problem any further. [3] give an extensive treatment of 15 popular error functions for time series extrapolation but is of little use for regression. A more relevant and extensive overview is given by Li and Zhao in [60] who discuss many practical metrics for performance estimation in general and propose a number of new ones. A more restricted and philosophical discussion is given in [44]. The previous references are mainly of theoretical nature. Empirical results on performance estimation are harder to find. One example is [22] who compare four different error functions used for neural network classification training.

Another topic that has been the subject of extensive research is that of multiobjective surrogate-based optimization (MOSBO). Surrogate methods

are widely used for the optimization of expensive functions [78]. While initially their use has been constrained to the single objective case, an increasing number of results are being reported in the multiobjective case. An example is the work on statistical improvement by Keane et al. [52] and ParEGO [56], the multiobjective version of the popular Efficient Global Optimization (EGO) approach [51]. Another example is the application to parameter optimization of earth system models in [76], for crashworthiness design optimization in [99], and for thin wall structure optimization in [86]. The well known NSGA-II algorithm [13] has also been extended to incorporate surrogate models [12, 97]. In this context some work has also been done on comparing different performance measures for use in MOSBO [54, 96]. [54] compare different performance criteria for improving metamodel based optimization. They also *"...recognize that in order to obtain desirable information or knowledge about a response surface, multiple performance measures taken in concert may be necessary."* Unfortunately they stop there and do not discuss the issue any further. Though the research into MOSBO is still young, an excellent overview of current research is already available in [57].

The contribution of the current work is that it deals with global surrogate modeling with iterative sampling and hyperparameter optimization. The goal is to generate a high fidelity global approximation using as few simulations as possible (replacement metamodeling) and minimizing user interaction. The dissertation takes an application perspective, and multiobjective optimization is considered on a higher, behavioral level ("What criteria should a model satisfy") versus a more model specific level ("How to generate a parsimonious neural network").

More concretely, the author stresses the importance of a critical analysis of performance estimation criteria and the associated trade-offs when generating surrogates (optimizing the hyperparameters). In particular, a founded choice of error function and target is often overlooked and performance estimation is done in a more ad hoc manner [7, 15], constrained to a single objective [32, 24, 23], or done a posteriori (after the model parameters have been fixed) to compare different models [71, 23]. While the implications and trade-offs of different performance criteria are well described in the statistics community (e.g., [3]), the resulting insights and possible solutions can use more visibility.

In addition we propose to model multi-output simulators simultaneously where this makes sense. Thus giving insight into the modeling trade-off between the outputs and avoiding multiple runs. An added benefit of this approach is the possibility of automatically selecting the best model type for each output. As [57] states *"Little is known about which types of model accord best with particular features of a landscape and, in any case, very little may be known to guide this choice."*. Thus an algorithm to automatically solve this problem is very useful [54]. This is also noticed by [97] who compare different surrogate models for approximating each objective during optimization.

They note that in theory their approach allows the use of a different model type for each objective. However, such an approach still requires an a priori model type selection and does not allow for dynamic switching of the model type or the generation of hybrids. We know of no other related work that tackles this issue.

In sum this paper takes a domain expert's point of view. By building on advances and established research in machine learning [50] and statistics [3] we attempt to further improve the global surrogate modeling process and make it more useful and accessible for an engineer. At the same time we hope to increase awareness of the issues involved.

2.1.6. Applications

This section presents some concrete illustrations of the ideas and concepts discussed previously. All tests were run using the SUrrogate MOdeling (SUMO) Toolbox which we first briefly discuss below.

2.1.6.1. The SUMO Toolbox

The SUMO Toolbox [39, 36] is an adaptive tool that integrates different modeling approaches and implements a fully automated, adaptive surrogate model construction algorithm. Given a simulation engine the toolbox produces a surrogate model within the time and accuracy constraints set by the user. Different plugins are supported: model types (rational functions, Kriging, splines, SVM, etc.), model parameter optimization algorithms (BFGS, EGO, simulated annealing, etc.), sample selection (random, error based, density based, etc.), and sample evaluation methods (local, on a cluster or grid). The behavior of each component is configurable through a central XML configuration file and components can easily be added, removed or replaced by custom implementations (see Figure 2.5).

The toolbox control flow is as follows: First, a small initial set of samples is chosen according to some experimental design (e.g., Latin hypercube, Box-Behnken, etc.). Based on this initial set, one or more surrogate models are constructed and their hyperparameters optimized according to a chosen hyperparameter optimization algorithm (e.g., BFGS, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), EGO, DIRECT, NSGA-II, etc.). Models are assigned a score based on one or more measures (e.g., cross validation, AIC, etc.) and the optimization continues until no further improvement is possible. The models are then ranked according to their score and new samples are selected based on the best performing models and the behavior of the response (the exact criteria depend on the sampling algorithm used). The hyperparameter optimization process is continued or restarted intelligently and the whole process repeats itself until one of the following three conditions is satisfied: (1) the maximum number of samples has been reached, (2) the maximum allowed time has been exceeded, or

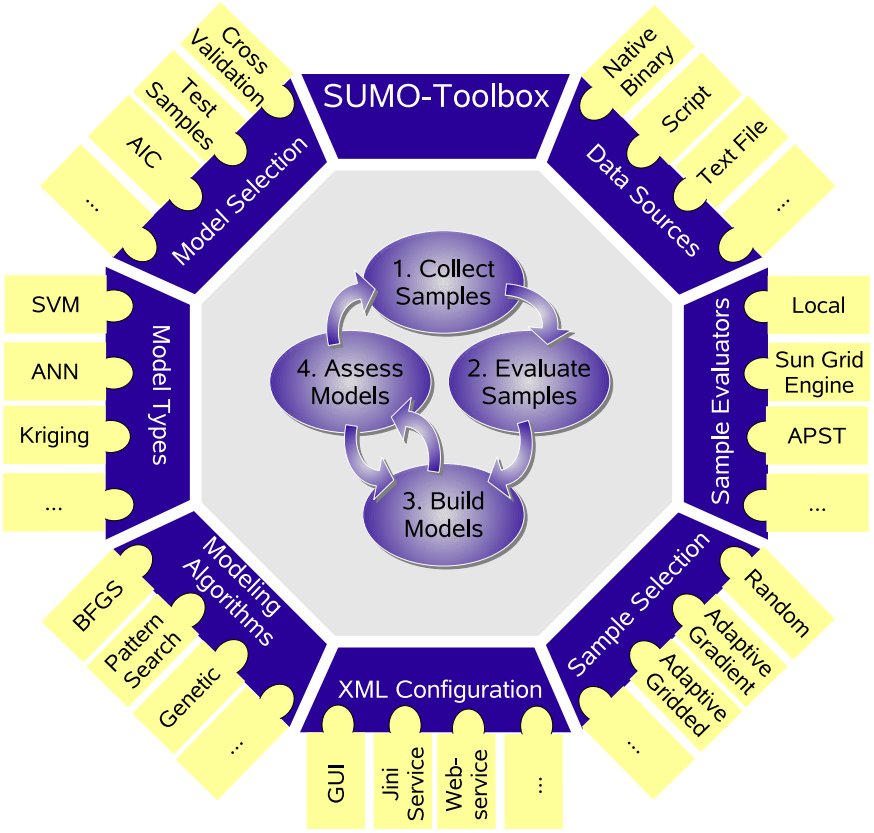


Figure 2.5.: SUMO Toolbox Plugins.

(3) the user required accuracy has been met. Also, the sample evaluation component runs in parallel with the other components (non-blocking) and not sequentially. The toolbox and all algorithms described here are available for download from `http://www.sumo.intec.ugent.be`.

2.1.6.2. Low Noise Amplifier (LNA)

Background We first consider a test case from electronics: a simple RF circuit, a narrow band Low Noise Amplifier (LNA) [58]. A LNA is the typical first stage of a receiver, having the main function of providing the gain needed to win the noise of subsequent stages, such as a mixer. In addition it has to give negligible distortion to the signal while adding as little noise as possible. We have extensively discussed the modeling of this system in [40, 41, 39]. For this paper we restrict ourselves to the 2D version and will use it to illustrate the use of multiple criteria in generating approximation models.

The input parameters are the (normalized) width of the MOSFET W_n and the normalized inductance L_{sn}. The output is the input noise current $\sqrt{\overline{i_{in}^2}}$ which previous results have shown to be the most difficult to model [40].

Experimental setup Previous experience with this function teaches us that this is a difficult function to model accurately with Kriging models (see [41]). Kriging models have difficulty reproducing the smooth surface of the input noise, they suffer from too many unwanted 'ripples' between the data points if a hold-out or cross validation measure is minimized. For this reason we consider two criteria. The first is the RRSE on a 20% min-max validation set, the second, a custom smoothness metric that penalizes a model if it produces ripples between data points.

The SUMO Toolbox (v6.1) was configured to use the Kriging [64] and NSGA-II [13] plugins. For the first run a fixed 7x7 factorial design was used, no additional sampling was performed. For the second run a density based sample selection algorithm was used that covers the design space evenly (previous tests showed it to give the best results with Kriging). Starting from a LHC design of 15 points together with the 4 corner points, the algorithm adds 15 points between each hyperparameter optimization iteration up to a maximum of 400.

For the first run NSGA-II was configured with a population size of 30 and run for a maximum of 250 generations. For the second run the maximum number of generations was set to 20, with the evolution continuing after each sampling iteration. Each individual in the population represents a Kriging model as a tuple (θ_1, θ_2) with θ_i the correlation parameter in log_{10} space ($\theta_i \in [-5, 3]$). The correlation function was set to Gaussian and a linear regression was used.

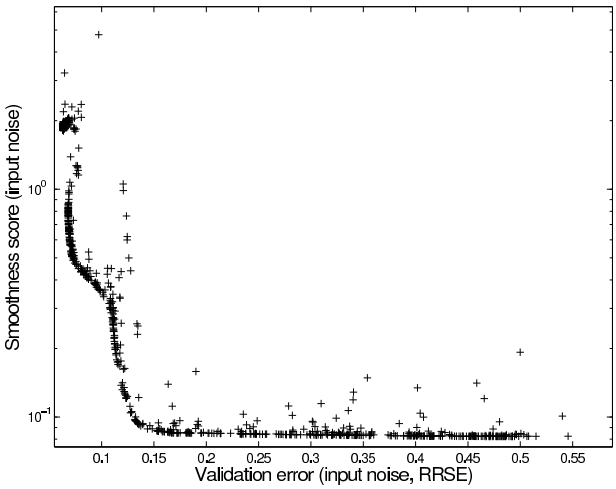


Figure 2.6.: Pareto search trace for the LNA problem (no sampling).

Results A plot of the full Pareto search trace for the first run (no sampling) is shown in Figure 2.6. As the figure shows there is a clear trade-off between the two objectives. This can also be seen from the plot of the model at each of the two extreme points (Figure 2.7). Given these results a domain expert now has the flexibility to browse though the front and select the most suitable model.

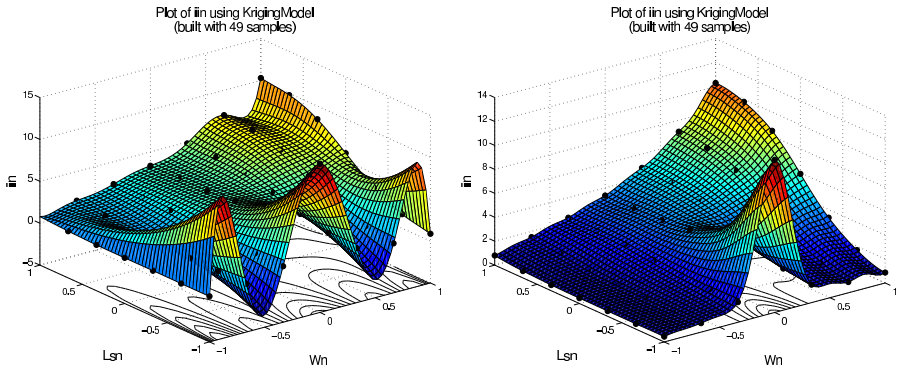
When sample selection is enabled the optimal Pareto set changes as more data becomes available. The successive Pareto fronts at the start of each sampling iteration are shown in Figure 2.8[3]. The figure clearly shows how the front advances and the model quality improves as more data becomes available. In addition the trade-off in the front seems to decrease as the number of points increase. This should be expected since as the amount of data increases there is less uncertainty about the correct hyperparameter values and the agreement between both measures increases.

2.1.6.3. Automotive problem

The second example is an application from the automotive industry (see [32] for details) and illustrates the modeling of a multi-output system.

Background Today the early concept phase of a car body development process is marked by the optimal coordination of design specifications with the requirements on the mechanical behavior of the structure as well as on the feasibility. This planning process is repetitive for the same body

[3] A movie showing the evolution is available at http://sumolab.blogspot.com/



(a) minimal validation error. **(b)** minimal smoothness penalty.

Figure 2.7.: Plot of the models at the extreme Pareto points for the LNA problem (no sampling).

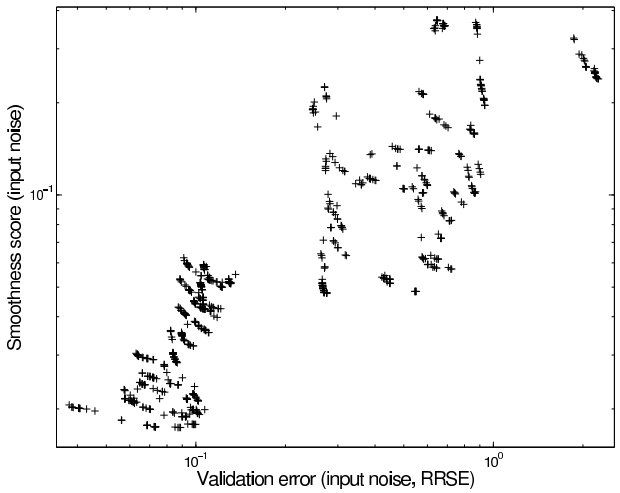


Figure 2.8.: Pareto search trace for the LNA problem (sampling enabled).

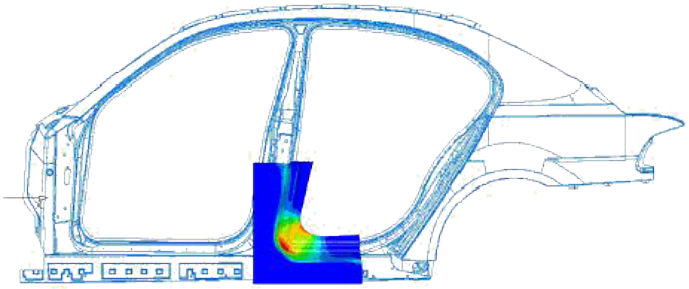


Figure 2.9.: B-pillar bottom of a side frame [32].

parts and the solution finding is carried out mostly by experience with an additional virtual tryout afterwards in order to improve the solution. The use of surrogate modeling can enable an early feasibility prediction of body parts.

The geometry of a B-pillar bottom of a side frame is shown in Figure 2.9. There you have a recurring feasibility challenge in sheet metal forming that can be explained by radii, depths and angles as experience shows. Which of these geometry parameters and in which combination they have an effect on the feasibility is, however, intuitively hard to predict. For simulation purposes the door entry area can be separated from the side frame by simple boundary conditions without major restrictions for the validity of the analysis but computing times considerably go down.

The entry angle α_1, opening angle α_2, frame depth h and entry radius r, have been chosen as geometry parameters (see [32] for details). In addition, for every geometry constellation the blank boundary was determined so that the forming result was optimal. Additional process parameters, like draw bead or blank holder forces, have not been used. So there were six parameters that have been taken into account. With these input quantities a sampling based on a LHC was created. Maximum scaled distances of the strain states to the forming limit curve and a maximum thinning limit respectively were chosen as output quantities indicating feasibility. The data sampling phase resulted in 1998 data points evaluated that were suitable for modeling. The overall target for this particular problem setting was to predict a given set of geometry parameters as feasible, i.e., to predict the existence of cracks (*cracking* output) or unacceptable thinning (*thinning* output).

Experimental setup Both outputs shall be modeled together using the ANN and LS-SVM plugins of the SUMO Toolbox. The ANN models are based on the Matlab Neural Network Toolbox and are trained with Levenberg Marquard backpropagation with Bayesian regularization [65, 29] (300

epochs). The topology and initial weights are determined by a GA. The LS-SVM models are based on the LS-SVMlab toolbox plugin [92] and the hyperparameters are searched in log_{10} space with $\sigma \in [-4, 4]$, $c \in [-5, 5]$ (an RBF kernel is used). The multiobjective algorithm used is the one implemented in the Matlab GADS toolbox which, in turn, is based on NSGA-II. The population size is set to 10. For comparison each output will be modeled separately as well (single objective).

In all cases the metric used to drive the hyperparameter optimization is the Average Relative Error (ARE) on 5-fold cross validation. For the single objective runs the timeout was 25 generations, for the multiobjective runs the timeout was 50 generations.

Results Figure 2.10 shows the final error curves after the SUMO Toolbox has terminated. A point is plotted for each time the toolbox finds a model that improves on the previous model. As can be seen from the figure, the ANN models clearly outperform the SVM models, especially for the *cracking* output. One could argue the poor performance of the LS-SVM models is due to poor hyperparameter optimization. However, this is not the case. For reference a brute force search of the hyperparameter landscape was conducted on a 50 by 60 grid. This is shown in Figure 2.11 (bounds in log_{10} scale, the white crosses show the area explored by the SUMO Toolbox). The minimum found through this search:

$$f_{cracking}(-0.1600, -1.4993) = 0.1348$$

$$f_{thinning}(0, -2.9996) = 0.0730$$

is comparable to the minimum found by the SUMO Toolbox:

$$f_{cracking}(-0.2173, 0.2978) = 0.1280$$

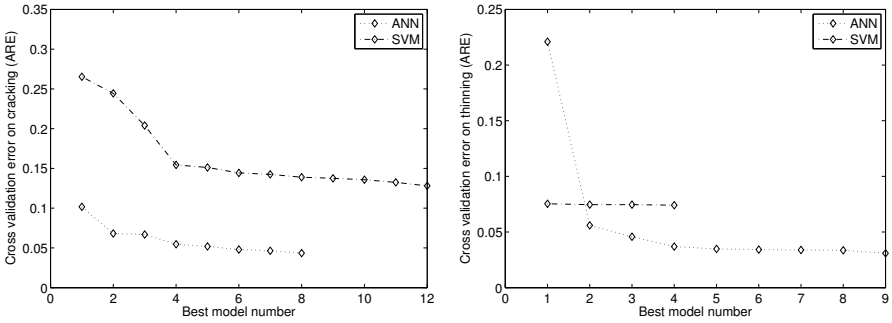
$$f_{thinning}(0.0423, 1.0948) = 0.0741$$

Thus the hyperparameter optimization is not to blame (remember that the cross validation procedure introduces some noise into the surface). A more extensive cross validation (15 folds) was also done on the final best model in each case (table 2.1). As can be seen, the accuracy remains unchanged.

The poor performance of SVMs in this case is in line with the author's previous experience. We found SVM models to require too much data when a non-linear, noise-free response needs to be fitted smoothly and accurately. In those cases, SVM models are very good at fitting the non-linear regions but generate unwanted 'ripples' in the regions where the response needs to be smooth or data is sparse. ANN models on the other hand, are able to adapt much better to the heterogeneity of the response. The sigmoid

Model type	*cracking*	*thinning*
ANN	0.0414	0.0325
LS-SVM	0.1305	0.0741

Table 2.1.: ARE on 15-fold cross validation of the final models (automotive example).



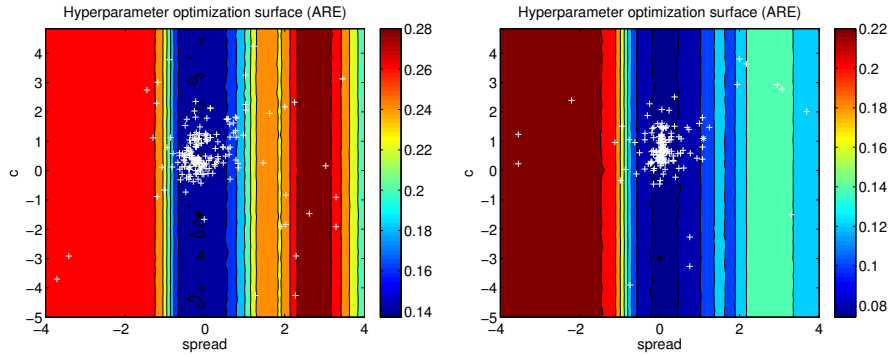
(a) *cracking*. **(b)** *thinning*.

Figure 2.10.: Model accuracies in the single objective case.

transfer functions allow for high non-linearity, while proper training (e.g., through the use of regularization) ensures a smooth fit in the sparse regions.

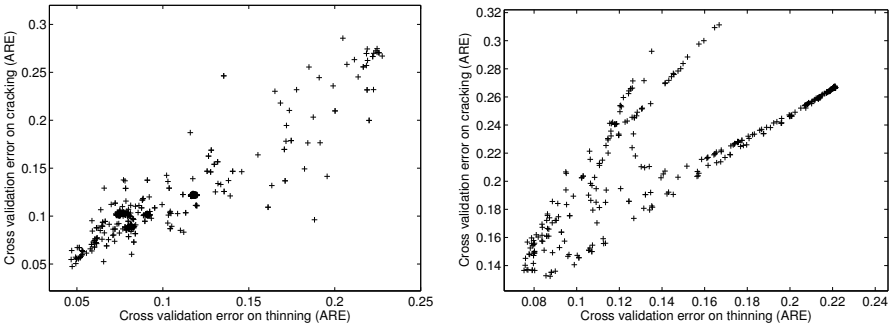
Figure 2.12 shows the full trace of the multiobjective hyperparameter optimization. In this case the model generation is driven by a 2-objective (= the cross validation score on each output) optimization algorithm. In both cases it is immediately clear that there is no real Pareto front, a single best model can be identified in each case. Thus this teaches us that there is a very strong correlation between both outputs and that good performance on one output, implies good performance on the other. This is actually to be expected since *cracking* and *thinning* are closely related (as can also be seen from Figure 2.11).

Of course this is not always the case (see for example [38]). It is not always clear how much the outputs are really correlated, or how much one quality metric influences another (in the case of multiple metrics). We argue that in those cases a direct multiobjective approach should be considered. It is guaranteed to give at least as much information as doing multiple single objective runs for about the same computational cost (which is still outweighed by the cost of the simulation). Also, it gives the engineer more flexibility and is a cleaner approach than manually combining the multiple objectives into a single formula.



(a) **(b)**

Figure 2.11.: SVM hyperparameter optimization surface. a) *cracking*. b) *thinning*.



(a) ANN. **(b)** SVM.

Figure 2.12.: Model accuracies in the multiobjective case.

2.1.6.4. Chemistry problem

Background This example and its description is taken from [45], where the authors describe the generation of an optimal ANN using a pattern search algorithm. We use this example to briefly illustrate the automatic model type selection per output. For a more extensive example see [37].

The chemical process under consideration describes methane-air combustion. The GRI 2.11 chemical mechanism containing 277 elementary chemical reactions among 49 species is used. The steady laminar flamelet equations [75] are often employed to describe the reaction-diffusion balance in non-premixed flames. The solutions to these equations provide temperature and mass fractions of all species in terms of two parameters. The mixture fraction z and the reaction progress variable c are used for this parametrization. The two responses are the temperature and the chemical source term of c, which can be viewed as a measure of heat release.

For the approximation 1000 data samples are available, half of which will be used for training, the other half to drive the hyperparameter optimization. Sample data were obtained by applying an acceptation-rejection method [80].

Experimental setup The heterogeneous evolution plugin of the SUMO Toolbox is used and configured with the following model types: RBF ANNs, LS-SVMs, and Rational functions. Together with the ensemble models (which result from a heterogeneous crossover, e.g., a crossover between a neural network and a rational function), this makes that 4 model types will compete to fit the data. The GA used is the NSGA-II based algorithm as implemented in the Matlab GADS toolbox. The population size of each model type is set to 10 and the evolution was run for 290 generations. A full discussion of the automatic model type selection algorithm is out of scope for this paper. Such details can be found in [35, 40]. The difference with the work discussed in [40] is that now the algorithms have been extended to the multiobjective case.

Results The full Pareto trace (enlarged for clarity) is shown in Figure 2.13. The figure shows that the LS-SVM models are best at fitting the temperature output, while fitting the chemical source term works best with a combination of models (ensemble). The ensembles turn out to consist of a combination of LS-SVM and RBFNN models. The rational functions turn out to perform very poorly on this data and are thus not shown on the (enlarged) figure. This trace can now also be used to generate a global ensemble of models (e.g., for uncertainty estimation).

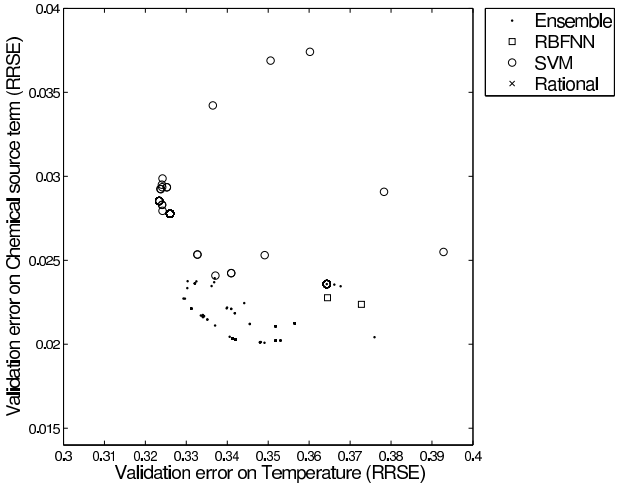


Figure 2.13.: Heterogeneous Pareto trace.

2.1.7. Summary and conclusion

The use of surrogate models to aid optimization, design exploration, sensitivity analysis, etc. has become standard practice among scientists and engineers alike. This work has concentrated on the construction of global surrogate models. A crucial problem of generating global surrogate models for a particular application (or any function approximation task for that matter), is agreeing upfront with the domain expert what criteria the final surrogate should satisfy. The problem is that each criterion (encompassing an error function, generalization estimator, and target value) involves a tradeoff between interpretability, accuracy, bias, and computational efficiency. Thus, for cases where this trade-off cannot be inferred from domain knowledge or application constraints the authors advocate a multiobjective approach to solving this problem should be considered. The advantage of a multiobjective approach is also that it allows multiple outputs to be modeled together, giving information about the tradeoff in the hyperparameter space. It further enables the generation of diverse ensembles and the application of an automatic model type selection algorithm. This enables each output to be automatically modeled with the most suitable model type. There is also some empirical evidence that the number of local optima can be reduced by converting multi-modal single-objective problems, into multiobjective ones [50]. If the same can be proven in machine learning it means the task of identifying good surrogate models can become easier through a multiobjective approach.

However, a disadvantage of the multiobjective approach is that as the number of dimensions (criteria/outputs) increases the solution space in-

creases exponentially [84]. Thus the search for the Pareto optimal set becomes harder, requires more search iterations, and the final set is more cumbersome for the practitioner to explore and understand. For costly simulation codes the extra computational effort is negligible, and good GUI tools can help a domain expert understand the relationships present in the Pareto-optimal set. However, for cheaper codes a trade off between simulation cost and modeling cost will have to be made. The poor scalability of non-dominated sorting algorithms above 4 dimensions is also an issue [57]. Luckily, algorithmic advances (e.g., [82, 68]) and gains in computational efficiency (e.g., [46]) continue to be made.

A disadvantage of the multiobjective approach versus the milestone approach is that the direct multiobjective approach takes all criteria into account straight away. This is not necessarily a problem but is not always the most computationally efficient. For example, in the case of adaptive sampling it makes no sense to check or enforce an (expensive) application specific constraint if only a few data points are available. The model first needs to mature by incorporating more data before undergoing more stringent checks. In this case the number of objectives varies dynamically and thus a scalarized multiobjective approach with a dynamically varying weighting parameter (as discussed in [49]) can be useful. Alternatively a cooling approach as done in [95] could be used.

Thus, naturally much work remains. First of all, while support for multiple criteria is already very useful, more research is needed on intuitive criteria. Ideally criteria should be easily formulated in language that a domain expert is comfortable with and fully understands. Fuzzy theory can be helpful in this respect. Besides researching the feasibility of fuzzy criteria more work still needs to be done on classic model selection methods and explore the relationship with a constraint based approach. This to fully understand the relationship between error function and generalization estimator, and how they impact the final response. A way to vary the criteria dynamically with the sample selection loop would also be useful as is the study of transductive learning [85]. A possible integration with domain partitioning methods (e.g., as done in [42]) is also promising.

Another area requiring further investigation is understanding how the iterative sample selection process influences the hyperparameter optimization landscape. There is a mutual dependency between the model type, hyperparameter optimization strategy, and sampling strategy (e.g., see [41]). The exact nature of this dependency depends on the model type. Determining how they interact and may be optimally combined is a topic of ongoing research. For the tests in this paper the authors have simply let the optimization continue from the previous generation. However, some initial tests have shown that an intelligent restart strategy can improve results. Knowledge of how the number and distribution of data points affects the hyperparameter surface (determined by some metric) would allow for a better tracking of the optimum, reducing the computational cost. The influ-

ence of noise and discrete variables on the hyperparameter optimization (e.g., neural network topology selection) also remains an issue.

In general, while some progress towards dynamic multiobjective optimization has been made [62, 43], this is a topic that current research in multiobjective surrogate modeling is only just coming to terms with [57]. Or as English pithily puts it: *"Optimization is easy, learning is hard (in the typical function)."* [21]

2.1.8. Bibliography

[1] N. Alexandrov, J. E. Dennis, R. M. Lewis, and V. Torczon. A trust region framework for managing the use of approximation models in optimization. *Structural Optimization*, 15:16–23, October 1998.

[2] D. Anderson and K. P. Burnham. *Model Selection and Multi-Model Inference*. Springer, 2003.

[3] J. S. Armstrong and F. Collopy. Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, 8(1):69–80, June 1992.

[4] P. L. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Machine Learning*, 48(1-3):85–113, 2002.

[5] R. R. Barton. Design of experiments for fitting subsystem metamodels. In *WSC '97: Proceedings of the 29th conference on Winter simulation*, pages 303–310, New York, NY, USA, 1997. ACM Press.

[6] A.. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive functions by surrogate. *Structural and Multidisciplinary Optimization*, 17(1):1–13, 1999.

[7] D. Busby, C. L. Farmer, and A. Iske. Hierarchical nonlinear approximation for experimental design and statistical data fitting. *SIAM Journal on Scientific Computing*, 29(1):49–69, 2007.

[8] H. Chen and S. Huang. A comparative study on model selection and multiple model fusion. In *Information Fusion, 2005 8th International Conference on*, volume 1, page 7pp., 25-28 July 2005.

[9] P-W. Chen, J-Y. Wang, and H-M. Lee. Model selection of SVMs using GA approach. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 3, pages 2035–2040, 25-29 July 2004.

[10] S. Conti and A. O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. research report no. 569/07, submitted to Journal of Statistical Planning and Inference. Technical report, Department of Probability and Statistics, University of Sheffield, 2007.

[11] J. De Geest, T. Dhaene, N. Faché, and D. De Zutter. Adaptive CAD-model building algorithm for general planar microwave structures. *IEEE Transactions on Microwave Theory and Techniques*, 47(9):1801–1809, Sep. 1999.

[12] K. Deb and P. Nain. An evolutionary multi-objective adaptive meta-modeling procedure using artificial neural networks. In Shengxiang Yang, Yew-Soon Ong, and Yaochu Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 297–322. Springer, 2007.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[14] V. K. Devabhaktuni and Q. Zhang. Neural network training-driven adaptive sampling algorithm. In *Proceedings of 30th European Microwave Conference, Paris, France*, volume 3, pages 222–225, 2000.

[15] V.K. Devabhaktuni, B. Chattaraj, M.C.E. Yagoub, and Q. Zhang. Advanced microwave modeling framework exploiting automatic model generation, knowledge neural networks, and space mapping. *IEEE Transactions on Microwave Theory and Techniques*, 51(7):1822–1833, Jul. 2003.

[16] M. Ding and R.I. Vemur. An active learning scheme using support vector machines for analog circuit feasibility classification. In *18th International Conference on VLSI Design*, pages 528–534, Jan. 2005.

[17] B. Efron. The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99:619–632, January 2004.

[18] A.A. El-Sallam, S. Kayhan, and A.M. Zoubir. Bootstrap and backward elimination based approaches for model selection. In *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis*, volume 1, pages 152–157, Sep. 2003.

[19] M. S. Eldred and D. M. Dunlavy. Formulations for surrogate-based optimization wiht data fit, multifidelity, and reduced-order models. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, Virginia*, pages AIAA–2006–7117, 2006.

[20] T.D. Eldred, M.S. Willcox, K.E. Robinson, and R. Haimes. Strategies for multifidelity optimization with variable dimensional hierarchical models. In *In Proceedings of the 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (2nd AIAA Multidisciplinary Design Optimization Specialist Conference), Newport, Rhode Island*, 2006.

[21] T.M. English. Optimization is easy and learning is hard in the typical function. *Proceedings of the 2000 Congress on Evolutionary Computation, 2000.*, 2:924–931 vol.2, 2000.

[22] T. Falas and A.-G. Stafylopatis. The impact of the error function selection in neural network-based classifiers. *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, 3:1799–1804 vol.3, 1999.

[23] H. Fang, M. Rais-Rohani, Z. Liu, and M.F. Horstemeyer. A comparative study of metamodeling methods for multiobjective crashworthiness optimization. *Computers and Structures*, 83(25-26):2121–2136, 2005.

[24] A. Farhang-Mehr and S. Azarm. Bayesian meta-modelling of engineering design simulations: a sequential approach with adaptation to irregularities in the response behaviour. *International Journal for Numerical Methods in Engineering*, 62(15):2104–2126, 2005.

[25] F. Fenicia, D. P. Solomatine, H. H. G. Savenije, and P. Matgen. Soft combination of local models in a multi-objective framework. *Hydrology and Earth System Sciences Discussions*, 4(1):91–123, 2007.

[26] J. E. Fieldsend. Multi-objective supervised learning. In J. Knowles, D. Corne, and K. Deb, editors, *Multiobjective Problem Solving from Nature From Concepts to Applications*, Natural Computing Series. Springer LNCS, 2008.

[27] J. E. Fieldsend and S. Singh. Pareto multiobjective nonlinear regression modelling to aid CAPM analogous forecasting. In *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, volume 1, pages 388–393, 12-17 May 2002.

[28] J. E. Fieldsend and S. Singh. Pareto evolutionary neural networks. *Neural Networks, IEEE Transactions on*, 16(2):338–354, March 2005.

[29] F.D. Foresee and M.T. Hagan. Gauss-Newton approximation to bayesian regularization. In *Proceedings of the 1997 International Joint Conference on Neural Networks*, pages 1930–1935, Jun. 1997.

[30] A.I.J. Forrester, N. W. Bressloff, and A. J. Keane. Optimization using surrogate models and partially converged computational fluid dynamics simulations. *Proceedings of the Royal Society*, 462:2177–2204, 2006.

[31] T. Furukawa, C. Jian K. Lee, and J. Michopoulos. Regularization for parameter identification using multi-objective optimization. In *Multi-Objective Machine Learning*, pages 125–149. 2006.

[32] M. Ganser, K. Grossenbacher, M. Schutz, L. Willmes, and T. Back. Simulation meta-models in the early phases of the product development process. In *Proceedings of Efficient Methods for Robust Design and Optimization (EUROMECH 07)*, 2007.

[33] T. Goel, R. Haftka, W. Shyy, and N. Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33:199–216, 2007.

[34] T. Goel, R.T. Haftka, and W. Shyy. Comparing error estimation measures for polynomial and kriging approximation of noise-free functions. *Journal of Structural and Multidisciplinary Optimization*, 28(5):429–442, June 2009.

[35] D. Gorissen. Heterogeneous evolution of surrogate models. Master's thesis, Master of AI, Katholieke Universiteit Leuven (KUL), 2007.

[36] D. Gorissen. Grid-enabled adaptive metamodeling and active learning for computer based design. In *Proceedings of The 22nd Canadian Conference on Artificial Intelligence (AI 2009), Kelowna*, volume LNCS 5549 of *Lecture Notes in Artificial Intelligence*, pages 266–269, May 2009.

[37] D. Gorissen, I. Couckuyt, K. Crombecq, and T. Dhaene. Pareto-based multi-output model type selection. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence (HAIS 2009), Salamanca, Spain*, pages 442–449. Springer - Lecture Notes in Artificial Intelligence, Vol. LNCS 5572, 2009.

[38] D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene. Pareto-based multi-output metamodeling with active learning. In *Proceedings of the 11th International Conference on Engineering Applications of Neural Networks (EANN 2009), London, England*, 2009.

[39] D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene. Sequential modeling of a low noise amplifier with neural networks and active learning. *Neural Computing and Applications*, 18(5):485–494, Jun. 2009.

[40] D. Gorissen, L. De Tommasi, J. Croon, and T. Dhaene. Automatic model type selection with heterogeneous evolution: An application to RF circuit block modeling. In *Proceedings of the IEEE Congress on Evolutionary Computation, WCCI 2008, Hong Kong*, 2008.

[41] D. Gorissen, L. De Tommasi, W. Hendrickx, J. Croon, and T. Dhaene. RF circuit block modeling via kriging surrogates. In *Proceedings of the 17th International Conference on Microwaves, Radar and Wireless Communications (MIKON 2008)*, 2008.

[42] H. Hamad and A. Al-Smadi. Space partitioning in engineering design via metamodel acceptance score distribution. *Engineering with Computers*, 23(3):175–185, 2007.

[43] I. Hatzakis and D. Wallace. Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1201–1208, New York, NY, USA, 2006. ACM.

[44] C. Hennig and M. Kutlukaya. Some thoughts about the design of loss functions. *REVSTAT - Statistical Journal*, 5:19–39, 2007.

[45] M. Ihme, A. L. Marsden, and H. Pitsch. Generation of optimal artificial neural networks using a pattern search algorithm: Application to approximation of chemical systems. *Neural Computation*, 20:573–601, 2008.

[46] M.T. Jensen. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. *Evolutionary Computation, IEEE Transactions on*, 7(5):503–515, Oct. 2003.

[47] Y. Jin. *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*. Springer, 2006.

[48] Y. Jin. Pareto-based multi-objective machine learning. In *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on*, pages 2–2, 17-19 Sept. 2007.

[49] Y. Jin, T. Okabe, and B. Sendhoff. Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: Why Does It Work and How? In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 1042–1049, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[50] Y. Jin and B. Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):397–415, May 2008.

[51] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[52] A. J. Keane. Statistical improvement criteria for use in multiobjective design optimization. *AIAA Journal*, 44(4):879–891, 2006.

[53] A. C. Keys and L. P. Rees. A sequential-design metamodeling strategy for simulation optimization. *Computers and Operational Research*, 31(11):1911–1932, 2004.

[54] A. C. Keys, L. P. Rees, and A. G. Greenwood. Performance measures for selection of metamodels to be used in simulation optimization. *Decision Sciences*, 33:31–58, 2007.

[55] J.P.C. Kleijnen, S. Sanchez, T. Lucas, and T. Cioppa. State-of-the-art review: A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, 17(3):263–289, 2005.

[56] J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[57] J. Knowles and H. Nakayama. Meta-modeling in multiobjective optimization. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 245–284. Springer-Verlag, Berlin, Heidelberg, 2008.

[58] T. H. Lee. *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge University Press, 2 edition, 2004.

[59] S. Lessmann, R. Stahlbock, and S.F. Crone. Genetic algorithms for support vector machine model selection. In *Proceedings of the International Joint Conference on Neural Networks, 2006. IJCNN '06.*, pages 3063–3069, 16-21 July 2006.

[60] X. Rong Li and Z. Zhao. Evaluation of estimation algorithms part I: incomprehensive measures of performance. *IEEE Transactions on Aerospace and Electronic Systems*, 42(4):1340–1358, October 2006.

[61] Y. Lin. *An Efficient Robust Concept Exploration Method and Sequential Exploratory Experimental Design*. PhD thesis, Georgia Institute of Technology, 2004.

[62] C-A. Liu and Y. Wang. Dynamic multi-objective optimization evolutionary algorithm. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 4, pages 456–459, 24-27 Aug. 2007.

[63] G.P. Liu and V. Kadirkamanathan. Learning with multi-objective criteria. *Fourth International Conference on Artificial Neural Networks*, pages 53–58, Jun 1995.

[64] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.

[65] D. MacKay. Bayesian model comparison and backprop nets. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 839–846. Morgan Kaufmann, Dec. 1992.

[66] Y. Matsuyama. Harmonic competition: a self-organizing multiple criteria optimization. *IEEE Transactions on Neural Networks*, 7(3):652–668, May 1996.

[67] M. Meckesheimer, A. J. Booker, R.R. Barton, and T.W. Simpson. Computationally inexpensive metamodel assessment strategies. *AIAA Journal*, 40(10):2053–2060, 2002.

[68] J. Mehnen, T. Wagner, and G. Rudolph. Evolutionary optimization of dynamic multi-objective test functions. In S. Cagnoni and L. Vanneschi, editors, *Digital Proceedings of the 3 Workshop Italiano di Vita Artificiale e della 2a Giornata di Studio Italiana sul Calcolo Evoluzionistico*. LabSEC Laboratorio Simulazioni di fenomeni Socio Economici Complessi, September 2006.

[69] I. Mierswa. Controlling overfitting with multi-objective support vector machines. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1830–1837, New York, NY, USA, 2007. ACM.

[70] A. M. Molinaro, R. Simon, and R. M. Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307, 2005.

[71] A. Mullur and A. Messac. Metamodeling using extended radial basis functions: a comparative approach. *Eng. with Comput.*, 21(3):203–217, 2006.

[72] A. O'Hagan. Bayesian analysis of computer code outputs: a tutorial. *Reliability Engineering and System Safety*, 91:1290–1300, 2006.

[73] Y-S. Ong, P.B. Nair, and K.Y. Lum. Max-min surrogate-assisted evolutionary algorithm for robust design. *Evolutionary Computation, IEEE Transactions on*, 10(4):392–404, Aug. 2006.

[74] Y-Y. Ou, C-Y. Chen, S-C. Hwang, and Y-J. Oyang. Expediting model selection for support vector machines based on data reduction. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 1, pages 786–791vol.1, 5-8 Oct. 2003.

[75] N. Peters. Laminar diffusion flamelet models in non-premixed turbulent combustion. *Progress in Energy and Combustion Science*, 10(3):319–339, 1984.

[76] A.R. Price, I.I. Voutchkov, G.E. Pound, N.R. Edwards, T.M. Lenton, and S.J. Cox. Multiobjective tuning of grid-enabled earth system models using a non-dominated sorting genetic algorithm (NSGA-II). In *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, pages 117–117, Dec. 2006.

[77] N. Queipo, R. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41:1–28, 2005.

[78] N. V. Queipo, C. J. Arévalo, and S. A. Pintos. The integration of design of experiments, surrogate modeling and optimization for thermoscience research. *Engineering with Computers*, 20(4):309–315, 2005.

[79] T. G. Robertazzi and S. C. Schwartz. An accelerated sequential algorithm for producing D-optimal designs. *Siam Journal on scientific Computing*, 10:341–358, Mar. 1989.

[80] R. Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc., New York, NY, USA, 1981.

[81] E. Sanchez, S. Pintos, and N.V. Queipo. Toward an optimal ensemble of kernel-based approximations with engineering applications. In *International Joint Conference on Neural Networks (IJCNN)*, volume 36, pages 247–261, 2006.

[82] N. Sangkawelert and N. Chaiyaratana. Diversity control in a multi-objective genetic algorithm. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 4, pages 2704–2711Vol.4, 8-12 Dec. 2003.

[83] M. J. Sasena, P. Y. Papalambros, and P. Goovaerts. Metamodeling sampling criteria in a global optimization framework. In *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, AIAA Paper 2000-4921.*, 2000.

[84] K. Sastry, D.E. Goldberg, and M. Pelikan. Limits of scalability of multiobjective estimation of distribution algorithms. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2217–2224Vol.3, 2-5 Sept. 2005.

[85] Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate gaussian process regression. In *NIPS*, pages 953–960, 2002.

[86] N. Mohamed Sheriffa, N.K. Guptab, R. Velmuruganc, and N. Shanmugapriyand. Optimization of thin conical frusta for impact energy absorption. *Thin-Walled Structures*, 46(6):653–666, 2008.

[87] T. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: Survey and recommendations. *Eng. Comput. (Lond.)*, 17(2):129–150, 2001.

[88] T. W. Simpson, V. Toropov, V. Balabanov, and F. A. C. Viana. Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come or not. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008 MAO, Victoria, Canada*, 2008.

[89] K. Smets, B. Verdonk, and E. M. Jordaan. Evaluation of performance measures for SVR hyperparameter selection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN2007)*, 2007.

[90] D. P. Solomatine and A. Ostfeld. Data-driven modelling : some past experiences and new approaches. *Journal of hydroinformatics*, 10(1):3–22, 2008.

[91] Thorsten Suttorp and Christian Igel. Multi-objective optimization of support vector machines. In *Multi-Objective Machine Learning*, pages 199–220. 2006.

[92] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing Co., Pte, Ltd., Singapore, 2002.

[93] D. J.J. Toal, N. W. Bressloff, and A. J. Keane. Kriging hyperparameter tuning strategies. *AIAA Journal*, 46(5):1240–1252, 2008.

[94] S. Tomioka, S. Nisiyama, and T. Enoto. Nonlinear least square regression by adaptive domain method with multiple genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 11(1):1–16, February 2007.

[95] C. Turner, R. Crawford, and M. Campbell. Multidimensional sequential sampling for NURBs-based metamodel development. *Engineering with Computers*, 23(3):155–174, 2007.

[96] K. K. Vasanth, K. Porkodi, and F. Rocha. Comparison of various error functions in predicting the optimum isotherm by linear and nonlinear regression analysis for the sorption of basic red 9 by activated carbon. *Journal of Hazardous materials*, 150:158–165, 2008.

[97] I. Voutchkov and A.J. Keane. Multiobjective Optimization using Surrogates. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture 2006. Proceedings of the Seventh International Conference*, pages 167–175, Bristol, UK, April 2006.

[98] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[99] L. Xingtao, L. Qing, Y. Xujing, Z. Weigang, and L. Wei. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization*, 35(6):561–569, June 2008.

[100] C. Yang, J. C. Meza, and L-W. Wang. A trust region direct constrained minimization algorithm for the kohn-sham equation. *SIAM Journal on Scientific Computing*, 29(5):1854–1875, 2007.

[101] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, September 1999.

[102] X. Yao and Y. Xu. Recent advances in evolutionary computation. *Journal of Computer Science and Technology*, 21(1):1–18, 2006.

[103] Y. Yuhong and A.R. Barron. An asymptotic property of model selection criteria. *Information Theory, IEEE Transactions on*, 44(1):95–116, Jan 1998.

[104] Z. Zhou, Y-S. Ong, and P.B. Nair. Hierarchical surrogate-assisted evolutionary optimization framework. In *Congress on Evolutionary Computation (CEC2004)*, volume 2, pages 1586–1593Vol.2, 19-23 June 2004.

[105] W. Zucchini. An introduction to model selection. *Journal of Mathematical Psychology*, 44:41–61, 2000.

2.2. ooDACE Toolbox

This section describes the theory of Kriging that is behind the ooDACE toolbox.

An associated journal paper and practical guide is found in Appendix B.

2.2.1. Introduction

The **ooDACE** [2, 1] toolbox is a versatile Matlab toolbox that implements the popular Gaussian process based Kriging surrogate models. Kriging is in particular popular for approximating (and optimizing) deterministic computer experiments [7, 14, 20]. The typical usage of the toolbox is to construct a Kriging model of a dataset obtained by (deterministic) computer simulations or measurements. Afterwards the Kriging surrogate can be fully exploited instead of the (more expensive) simulation code. The toolbox is aimed for solving complex applications (expensive simulation codes, physical experiments, ...) and for researching new Kriging extensions and techniques.

Section 2.2.2 discusses the key mathematical formulae of different types of Kriging and gives some insights into various properties.

2.2.2. Theory

First conceived by Danie Krige in geostatistics, Kriging, also known as Gaussian process, is a surrogate model to approximate deterministic noise-free data, and has proven to be very useful for tasks such as optimization [7], design space exploration, visualization, prototyping, and sensitivity analysis [20]. A thorough mathematically treatment of Kriging is given in [15, 3]. The popularity of Kriging has generated a large body of research, including several extensions to Kriging to handle different problem settings, e.g. by adding gradient information in the prediction [12], or by approximating stochastic simulations [17].

In the remainder of this section we will give a brief overview of each type of Kriging available in the ooDACE toolbox.

2.2.2.1. Kriging

Basically, Kriging is a two-step process: first a regression function $f(\mathbf{x})$ is constructed based on the data, and, subsequently, a Gaussian process Z is constructed through the residuals.

$$Y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}), \tag{2.15}$$

where $f(\mathbf{x})$ is a regression (or trend) function and Z is a Gaussian process with mean 0, variance σ^2 and a correlation matrix Ψ.

Depending on the form of the regression function Kriging has been prefixed with different names. *Simple* Kriging assumes the regression function to be a known constant, i.e., $f(\mathbf{x}) = 0$. A more popular version is *ordinary* Kriging, which assumes a constant but unknown regression function $f(\mathbf{x}) = \alpha_0$. Though, other, more complex, trend functions are possible such as linear or quadratic polynomials. In general, *universal* Kriging treats the trend function as a multivariate polynomial, namely,

$$f(\mathbf{x}) = \sum_{i=1}^{p} \alpha_i b_i(\mathbf{x}), \tag{2.16}$$

where $b_i(\mathbf{x})$ are $i = 1 \dots p$ basis functions (e.g., the power base for a polynomial) and $\alpha = (\alpha_1, \dots, \alpha_p)$ denotes the coefficients. The idea is that the regression function captures the largest variance in the data (the general trend) and that the Gaussian process interpolates the residuals. However, selecting the correct regression function is a difficult problem, hence, the regression function is often chosen constant (=ordinary Kriging).

Consider a set of n samples, $X = \{\mathbf{x}^1, \dots, \mathbf{x}^n\}$ in d dimensions and associated function values, $\mathbf{y} = \{y^1, \dots, y^n\}$. Essentially, the regression part is encoded in the $n \times p$ model matrix F,

$$F = \begin{pmatrix} b_1(\mathbf{x^1}) & \cdots & b_p(\mathbf{x^1}) \\ \vdots & \vdots & \vdots \\ b_1(\mathbf{x^n}) & \cdots & b_p(\mathbf{x^n}) \end{pmatrix},$$

while the stochastic process is mostly defined by the $n \times n$ correlation matrix Ψ,

$$\Psi = \begin{pmatrix} \psi(\mathbf{x}^1, \mathbf{x}^1) & \dots & \psi(\mathbf{x}^1, \mathbf{x}^n) \\ \vdots & \ddots & \vdots \\ \psi(\mathbf{x}^n, \mathbf{x}^1) & \dots & \psi(\mathbf{x}^n, \mathbf{x}^n) \end{pmatrix},$$

where $\psi(\cdot, \cdot)$ is the correlation function. $\psi(\cdot, \cdot)$ is parametrized by a set of hyperparameters θ, which are identified by Maximum Likelihood Estimation (MLE), see Section 2.2.2.7. Subsequently, the prediction mean and prediction variance of Kriging are derived, respectively, as,

$$\mu(\mathbf{x}) = M\alpha + r(\mathbf{x}) \cdot \Psi^{-1} \cdot (\mathbf{y} - F\alpha), \tag{2.17}$$

$$s^2(\mathbf{x}) = \sigma^2 \left(1 - r(\mathbf{x})\Psi^{-1} r(\mathbf{x})^T + \frac{(1 - F^T\Psi^{-1} r(\mathbf{x})^T)}{F'\Psi^{-1}F}\right), \tag{2.18}$$

where $M = \begin{pmatrix} b_1(\mathbf{x}) & b_2(\mathbf{x}) & \dots & b_p(\mathbf{x}) \end{pmatrix}$ is the model matrix of the predicting point $\mathbf{x}$, $\alpha = (F^T\Psi^{-1}F)^{-1}F^T\Psi^{-1}\mathbf{y}$ is a $p \times 1$ vector denoting the coefficients of the regression function, determined by Generalized Least Squares (GLS), and $r(\mathbf{x}) = \begin{pmatrix} \psi(\mathbf{x}, \mathbf{x}^1) & \dots & \psi(\mathbf{x}, \mathbf{x}^n) \end{pmatrix}$ is an $1 \times n$ vector of correlations between the point $\mathbf{x}$ and the samples X. The process variance σ^2 is given by $\frac{1}{n}(\mathbf{y} - F\alpha)^T\Psi^{-1}(\mathbf{y} - F\alpha)$.

Note that Kriging, as formulated here, is an interpolation technique. This is easily seen by substituting the i^{th} sample point $\mathbf{x}^i$ in Equation (2.17) and considering that $r(\mathbf{x}^i)$ is the i^{th} column of Ψ, hence, $r(\mathbf{x}^i) \cdot \Psi^{-1}$ is an unit vector e_i with a 1 at the i^{th} position,

$$\mu(\mathbf{x}^i) = M\alpha + e_i \cdot (\mathbf{y} - F\alpha) = M\alpha + y_i - M\alpha = y^i. \tag{2.19}$$

Partial derivatives of the prediction mean with respect to a test point $\mathbf{x}$ are given by,

$$\frac{\partial \mu(\mathbf{x})}{\partial x_i} = \frac{\partial M}{\partial x_i}\alpha + \frac{\partial r(\mathbf{x})}{\partial x_i}\Psi^{-1}(\mathbf{y} - F\alpha), \tag{2.20}$$

where the matrix $\frac{\partial M}{\partial x_i}$ and vector $\frac{\partial r(\mathbf{x})}{\partial x_i}$ are obtained by taking the derivatives of the individual entries respectively, namely,

$$\frac{\partial M}{\partial x_i} = \begin{pmatrix} \frac{\partial b_1(\mathbf{x})}{\partial x_i} & \frac{\partial b_2(\mathbf{x})}{\partial x_i} & \dots & \frac{\partial b_p(\mathbf{x})}{\partial x_i} \end{pmatrix},$$

$$\frac{\partial r(\mathbf{x})}{\partial x_i} = \begin{pmatrix} \frac{\partial \psi(\mathbf{x}, \mathbf{x}^1)}{\partial x_i} & \dots & \frac{\partial \psi(\mathbf{x}, \mathbf{x}^n)}{\partial x_i} \end{pmatrix}.$$

2.2.2.2. Co-Kriging

Co-Kriging exploits the correlation between fine and coarse model data to enhance the prediction accuracy. The ooDACE toolbox uses the autoregressive co-Kriging model of Kennedy et al. [11]. Consider two sets of samples, $X_c = \{\mathbf{x}_c^1, \dots, \mathbf{x}_c^{n_c}\}$ and $X_e\{\mathbf{x}_e^1, \dots, \mathbf{x}_e^{n_e}\}$, with dimension d obtained from the low-fidelity (cheap) and high-fidelity (expensive) simulator, respectively. The associated function values are denoted by $\mathbf{y}_c = \{y_c^1, \dots, y_c^{n_c}\}$ and $\mathbf{y}_e = \{y_e^1, \dots, y_e^{n_e}\}$.

Creating a co-Kriging model can be interpreted as constructing two Kriging models in sequence. First a Kriging model $Y_c(\mathbf{x})$ of the coarse data $(X_c, \mathbf{y}_c)$ is constructed. Subsequently, the second Kriging model $Y_d(\mathbf{x})$ is constructed on the residuals of the fine and coarse data $(X_e, \mathbf{y}_d)$, where $\mathbf{y}_d = \mathbf{y}_e - \rho \cdot \mu_c(X_e)$. The parameter ρ is estimated as part of the MLE of the second Kriging model. Note that the configuration (the choice of the correlation function, regression function, etc.) of both Kriging models can be adjusted separately for the coarse data and the residuals, respectively.

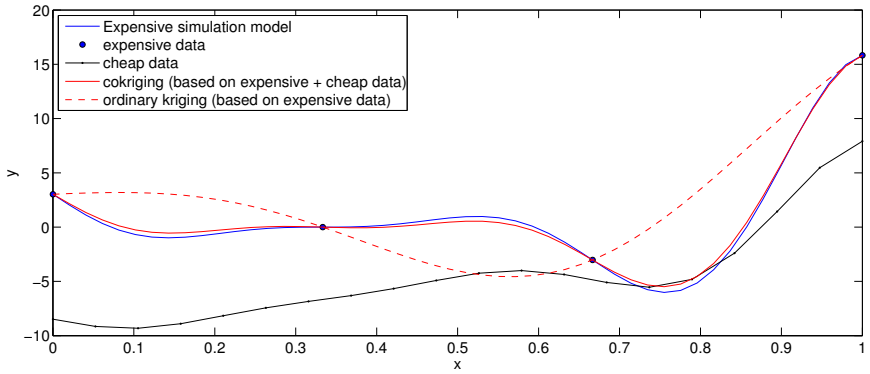


Figure 2.14.: Kriging and co-Kriging applied to a 1-dimensional mathematical example. Co-Kriging interpolates the fine data and is further corrected by the coarse data.

The resulting co-Kriging interpolant is then defined similarly as Equation (2.17),

$$\mu(\mathbf{x}) = M\alpha + r(\mathbf{x}) \cdot \Psi^{-1} \cdot (\mathbf{y} - F\alpha), \tag{2.21}$$

where the block matrices M, F, $r(\mathbf{x})$ and Ψ can be written in function of the two separate Kriging models $\mu_c(\mathbf{x})$ and $\mu_d(\mathbf{x})$:

$$r(\mathbf{x}) = \begin{pmatrix} \rho \cdot \sigma_c^2 \cdot r_c(\mathbf{x}) & \rho^2 \cdot \sigma_c^2 \cdot r_c(\mathbf{x}, X_f) + \sigma_d^2 \cdot r_d(\mathbf{x}) \end{pmatrix},$$

$$\Psi = \begin{pmatrix} \sigma_c^2 \cdot \Psi_c & \rho \cdot \sigma_c^2 \cdot \Psi_c(X_c, X_f) \\ \mathbf{0} & \rho^2 \cdot \sigma_c^2 \cdot \Psi_c(X_f, X_f) + \sigma_d^2 \cdot \Psi_d \end{pmatrix},$$

where $(F_c, \sigma_c, \Psi_c, M_c)$ and $(F_d, \sigma_d, \Psi_d, M_d)$ are matrices obtained from the Kriging models $Y_c(\mathbf{x})$ and $Y_d(\mathbf{x})$, respectively (see Section 2.2.2.1). In particular, σ_c^2 and σ_d^2 are process variances, while $\Psi_c(\cdot,\cdot)$ and $\Psi_d(\cdot,\cdot)$ denote correlation matrices of two datasets using the optimized $\theta_1 \ldots, \theta_n$ parameters and correlation function of the Kriging models $Y_c(\mathbf{x})$ and $Y_d(\mathbf{x})$, respectively. An illustration of co-Kriging on an one-dimensional example is shown in Figure 2.14.

2.2.2.3. Blind Kriging

As the actual full behavior of the response is unknown it is often hard to choose which trend function $f(\mathbf{x})$ to use for a given problem. Feature selection methods [5] offer the possibility to identify the most plausible interactions occurring in the data. Blind Kriging [10] extends Kriging with a Bayesian feature selection method.

The goal of blind Kriging is to efficiently determine the basis functions b_i (features) that captures the most variance in the sample data. To that end, a set of candidate functions is considered from which to choose from. In the ideal case the sample data is almost fully represented by the chosen trend function and the stochastic process $Z(\mathbf{x})$ has little or no influence.

Consider an existing Kriging model $Y(\mathbf{x})$, e.g., with a constant regression function (ordinary Kriging). The idea is to select new features to be incorporated in the regression function of this Kriging model, taking into account features that are already part of the regression function of this Kriging model. To that end, the whole set of candidate functions c_i is used to fit the data in a linear model, i.e.,

$$g(\mathbf{x}) = \sum_{i=1}^{p} \alpha_i b_i(\mathbf{x}) + \sum_{i=1}^{t} \beta_i c_i(\mathbf{x}),$$

where t denotes the number of candidate functions.

The first part of this equation is the regression function of Kriging and, hence, the coefficients α have already been determined independently of $\beta = (\beta_1, \ldots, \beta_t)$. The estimation of β provides a relevance score of the candidate features. A frequentist estimation of β (e.g., least-squares solution) would be a straightforward approach to rank the features. However, this is not always possible as the number of candidate features is often higher than the number of samples available. For instance, considering all possible interactions up to the quadratic effect in four dimensions the number of candidate features is $t = 3^4 = 81$. To that end, a Gaussian Prior distribution is introduced for β,

$$\beta \sim \mathcal{N}(\mathbf{0}, \tau^2 R), \tag{2.22}$$

where $R = U^{-1}\Psi(U^{-1})^T$ and U is the model matrix, namely, a design matrix with t rows. Furthermore, the choice of correlation functions is restricted to the product correlation form,

$$\psi(\mathbf{x}, \mathbf{x}') = \prod_{j=1}^{d} \psi_j(|x_j - x_j'|), \tag{2.23}$$

the variance-covariance matrix R can be constructed independently of the number of dimensions,

$$R_j = U_j^{-1}\psi_j(U_j^{-1})^T,$$

where U_j is the model matrix of the samples for factors $j = 1 \ldots d$. Thus the number of considered features can be chosen per dimension and afterwards the full matrix R is obtained by taking the Kronecker product,

$$R = \bigotimes_{j=1}^{d} R_j.$$

While the Bayesian variable selection is able to handle arbitrarily high-order effects, the matrix R grows quite rapidly. Hence, it may be useful to consider the special case where only linear effects, quadratic effects and two-factor interactions are identified. The total set of candidate functions is then defined by $c_i(\mathbf{x})$, where $i = 1 \ldots t = 2d^2$. Note that t does not scale exponentially as above, but still the matrix R would already require $(t+1) \times (t+1)$ $(> 4d^4)$ entries.

Let U_j be 3×3 orthogonal polynomial coded [21] matrices, then

$$R_j = U_j^{-1}\psi_j(U_j^{-1})^T = U_j^T\psi_j U_j = \begin{pmatrix} 3+4\psi_j(1)+2\psi_j(2) & 0 & -\sqrt{2}(\psi_j(1)-\psi_j(2)) \\ 0 & 3(1-\psi_j(2)) & 0 \\ -\sqrt{2}(\psi_j(1)-\psi_j(2)) & 0 & 3-4\psi_j(1)+\psi_j(2) \end{pmatrix}, \quad (2.24)$$

this requires scaling of the sample data to the interval $[1, 3]$. The encoded samples for linear and quadratic effects are then, respectively, defined by,

$$x_{j,l} = \frac{\sqrt{3}}{\sqrt{2}}(\mathbf{x}_j - 2), \quad (2.25)$$

$$x_{j,q} = \frac{1}{\sqrt{2}}(3(\mathbf{x}_j - 2)^2 - 2), \quad (2.26)$$

for $j = 1 \ldots d$, where $\mathbf{x}_j$ denotes the j^{th} column of X. Other candidate terms can be constructed from these basic effects, e.g., the linear-quadratic effect between x_{15} and x_6 is represented by $b_i = x_{15,l} \cdot x_{6,q}$ for a particular i. As x_j takes on the values 1, 2 and 3 the column lengths of $x_{j,l}$ and $x_{j,q}$ will be $\sqrt{3}$.

While there is some (negative) correlation between mean and quadratic effects (see Equation 2.24), [8, 9] propose to only use the information of the diagonal of R_j. Normalizing to the mean $3 + 4\psi_j(1) + 2\psi(2)$ (first entry of the diagonal of R_j) the variance-covariance matrix R can be expressed as follows. For ease of notation let $\psi(\mathbf{x})$ be a vector of length d,

$$\psi(\mathbf{x}) = \begin{pmatrix} \psi_1(\mathbf{x}) \\ \vdots \\ \psi_d(\mathbf{x}) \end{pmatrix},$$

the vectors $\mathbf{r_l}$ and $\mathbf{r_q}$ of length d are then defined by,

$$\mathbf{r}_l = \frac{3 - 3\psi(2)}{3 + 4\psi(1) + 2\psi(2)}, \tag{2.27}$$

$$\mathbf{r}_q = \frac{3 - 4\psi(1) + \psi(2)}{3 + 4\psi(1) + 2\psi(2)}, \tag{2.28}$$

finally let $\mathbf{l_i}$ be the vector where element $l_{i,j} = 1$ if β_i includes the linear effect of factor j and 0 otherwise. In addition, define $\mathbf{q_i}$ as the vector where element $q_{i,j} = 1$ if β_i includes the quadratic effect of factor j and 0 otherwise. Then the diagonal matrix R is defined as,

$$R = \begin{pmatrix} \mathbf{r}_l^{l_1} \cdot \mathbf{r}_q^{q_1} & 0 & \dots & 0 \\ 0 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{r}_l^{l_{t+1}} \cdot \mathbf{r}_q^{q_{t+1}} \end{pmatrix}. \tag{2.29}$$

Note that, as the correlations between mean and quadratic effects have been dropped from Equation (2.24) the matrix R is in fact an estimation of the real correlation matrix.

Having constructed the covariance matrix R by any means explained above, the posterior of β is estimated by,

$$\hat{\beta} = \frac{\tau^2}{\sigma^2} R M_c^T \Psi^{-1} (\mathbf{y} - M \cdot \mathbf{a}), \tag{2.30}$$

$$var(\hat{\beta}) = \tau^2 (R - \frac{\tau^2}{\sigma^2} R M_c^T \Psi^{-1} M_c R), \tag{2.31}$$

where M_c is the $n \times (t+1)$ model matrix of all candidate variables, M is the model matrix of all currently chosen variables and Ψ is the correlation matrix of the samples.

The coefficients $\hat{\beta}$ obtained through this Bayesian variable ranking method quantifies the importance of the associated candidate feature with respect to the data. There are several heuristics proposed to identify the best set of variables to approximate the data. Originally [9], the feature selection consisted of a greedy forward selection procedure, iteratively adding candidate variables with highest standardized coefficients. In blind Kriging [10] the standardized coefficient is replaced with the absolute value of $\hat{\beta}$, delivering similar results while easier to compute. Note that the first term of Equation 2.30 is a constant and does not influence the end results, i.e., $\frac{\tau^2}{\sigma^2}$ is set to 1.

The advantage of choosing this Bayesian variable selection method over other techniques is the tight coupling with Kriging's correlation matrix Ψ, thus taking advantage of already available information. Moreover, this variable selection method incorporates the important variable selection

principles - effect hierarchy[4] and effect heredity[5] [6] - in the prior belief of β. In other words, the chosen features should form a simple and interpretable regression function.

In summary, constructing blind Kriging models can be seen as a two stage process. In the first phase an ordinary Kriging model, namely, a Kriging model with a constant regression function, is constructed and θ parameters are estimated. In a second phase the regression function of this initial Kriging model is extended with promising features according to the estimated $\hat{\beta}$ coefficients, generating a series of intermediate Kriging models. When these intermediate Kriging models stop improving on the leave-one out cross validation prediction error, the search is halted (though a look-ahead of n steps can be used to avoid local optima). The current best set of features is then chosen to construct the final blind Kriging model, re-estimating the θ parameters.

2.2.2.4. Stochastic Kriging

While the interpolation property (see Equation (2.19)) of Kriging is advantageous for many (deterministic) simulation problems, it might produce undesired results when dealing with stochastic simulations and/or in the presence of noise. Stochastic Kriging [17] extends Kriging for approximation instead of interpolation, also known as regression Kriging. To that end, the noise is modeled as a separate Gaussian process $\xi(\mathbf{x})$ with mean 0, and covariance matrix $\sum$,

$$\xi \sim \mathcal{GP}(0, \Sigma).$$

The stochastic Kriging predictor then becomes,

$$\hat{y}(\mathbf{x}) = M\alpha + r(\mathbf{x}) \cdot (\Psi + \frac{1}{\sigma^2}\sum)^{-1} \cdot (\bar{\mathbf{y}} - F\alpha), \tag{2.32}$$

where $\frac{1}{\sigma^2}\sum$ is a matrix resembling noise-to-signal ratios and $\bar{\mathbf{y}}$ is a vector containing the average function values of the repeated simulations for each sample. Note that if the entries of Σ are zero (no noise) the formula is the same as Equation (2.17) and will interpolate the data exactly like universal Kriging.

Depending on the type and distribution of noise the covariance matrix $\sum$ has different forms. In stochastic simulation Σ can be created based on repeated simulations and, in its simplest form, can be defined as,

[4] Effect hierarchy denotes that low order effects (e.g., individual variables) should be chosen before high order effects (e.g., interactions of variables)

[5] Effect heredity states that an effect cannot be important until its parent effect is also important

$$\Sigma = \begin{pmatrix} var(\mathbf{y}^1) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & var(\mathbf{y}^n) \end{pmatrix},$$

where $var(\mathbf{y}^i)$ is the variance between the repeated simulations of data point i.

However, stochastic Kriging (or regression Kriging) is also useful for deterministic (but noisy) simulation problems (or measurements), where repeated simulations are not available. Assuming the noise is homogeneous distributed across the input space the matrix Σ consists of a scalar value (10^λ) on its diagonal, i.e., $\sum = 10^\lambda I_n$, where I_n is the $n \times n$ identity matrix. The variable λ (amount of noise) is estimated as part of the likelihood optimization of Kriging. Logically, in case of heterogeneous noise the matrix Σ has different values on the diagonal, which introduces more variables in the likelihood optimization problem.

2.2.2.5. Miscellaneous

In addition to the distinct types of Kriging models the ooDACE toolbox also incorporates several smaller extensions and useful tools. A number of those tools are described in this section.

Re-interpolation of the prediction variance When using stochastic Kriging (or regression Kriging) the prediction variance is not zero anymore in the sample points. However, this is actually a desired property for many algorithms to achieve some form of space-fillingness (e.g., maximizing the expected improvement or prediction variance). To that end, Forrester et al. [4] suggest a re-interpolation of the prediction variance. Basically, this is done by ignoring the Σ matrix in the respected formulae. In particular, the process variance σ^2 and the prediction variance formulations are taken from standard Kriging, see Equation (2.18), which differ only from stochastic Kriging with respect to the covariance matrix Σ.

Leave-one-out cross validation The leave-one-out cross validation (or cross validated prediction error; cvpe) score [19, 10] can be efficiently calculated as follows:

$$H = F(F^T F)^{-1} F^T,$$

$$\mathbf{d} = \mathbf{y} - F\alpha,$$

$$cvpe = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{(\Psi^{-1})_{i,:} \cdot \left(\mathbf{d} + H_{:,i} \cdot \frac{\mathbf{d}_i}{1 - H_{ii}} \right)}{(\Psi^{-1})_{ii}} \right)^2, \tag{2.33}$$

where $A_{i,:}$, $A_{:,i}$ denote the i^{th} row and column, respectively, for a matrix A while A_{ii} is the i^{th} entry on the diagonal.

Integrated mean square error The integrated mean square error (imse) [14] is another criterion to measure the accuracy of a Kriging model and is defined as,

$$imse = \int_A s^2(\mathbf{x}) dA, \tag{2.34}$$

where A denotes the input domain. Naturally, it is impossible to evaluate this integral analytically, hence, approximation methods should be used such as trapezoidal numerical integration or Monte Carlo sampling.

Error on a test set A safe way to measure the accuracy of a surrogate model is to use an independent test set $X_{test} = \{\mathbf{x}^1_{test}, \ldots, \mathbf{x}^{n_{test}}_{test}\}$ with associated function values $y^1_{test}, \ldots, y^{n_{test}}_{test}$ (if available).

$$error = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \left(\mu(\mathbf{x}^i_{test}) - y^i_{test}\right)^2. \tag{2.35}$$

Robustness criterion Siem et al. [16] propose the robustness-criterion to measure the stability of an ordinary Kriging model with respect to simulation errors. Let $\gamma = \Psi^{-1} \cdot (\mathbf{y} - F\alpha)$ then the absolute and relative robustness criterion are defined as,

$$rc_{absolute} = |\gamma|_2^2, \tag{2.36}$$

$$rc_{relative} = |\gamma|_2^2 / |\gamma|_\infty. \tag{2.37}$$

2.2.2.6. Correlation functions

Arguably, the choice of correlation function is crucial to create an accurate Kriging surrogate model, whether it is universal Kriging, co-Kriging, stochastic Kriging, etc. A popular class of *stationary* correlation functions is defined by,

$$\psi(\mathbf{x}, \mathbf{x}') = \exp\left(-\sum_{i=1}^{d} \theta_i |x_i - x'_i|^p\right).$$

Note that these correlation functions only depend on the distance between the two points $\mathbf{x}$ and $\mathbf{x}'$. The smaller the distance between two points, the higher the correlation and, hence, the more the prediction of one point is influenced by the other, i.e., their function values are closer together. Similarly, if the distance increases the correlation drops to zero. The rate and manner at which this happens is governed by several parameters. The parameter p determines the initial drop in correlation as distance increases,

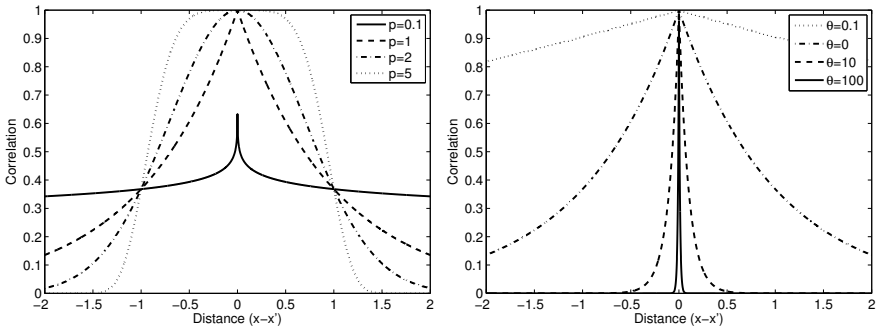


Figure 2.15.: The one-dimensional Gaussian correlation functions: a) with varying parameter p for $\theta = 1$) b) with varying parameter θ for $p = 1$.

see Figure 2.15a. Often p is set to two (=the Gaussian correlation function) which assumes that the data represents a smooth, continuous surface. A lower value of p, e.g., $p = 1$, is more suitable for a more rough (sharp/erratic) response as it permits a more substantial difference in function values for points close together.

The second set of parameters, $\theta_1, \ldots, \theta_d$, describes the influence sphere of a point on nearby points for each dimension, i.e., how fast the correlation drops to zero, see 2.15b. Usually, p is set fixed while the parameters $\theta_1, \ldots, \theta_d$ are identified using Maximum Likelihood Estimation (MLE). The MLE's of $\theta_1, \ldots, \theta_d$ are useful as they describe the amount of variation in each dimension i. A high value of θ_i means points have less influence on each other and, thus, similar points in the input space can have a very different response value (highly non-linear behavior in dimension i). On the other hand, a low value of θ_i indicates that a point is correlated with points that are farther away (a more linear behavior).

The ooDACE toolbox offers three instances of this well-known class of correlation functions,

- The Gaussian correlation function ($p = 2$, *corrgauss*)
- The exponential correlation function ($p = 1$, *correxp*)
- A generic version where p is included in the likelihood optimization (*corrgaussp*)

While, arguably, these are the most frequently used correlation functions - and in particular the Gaussian correlation function - to solve engineering problems, the Matérn class of correlation functions [18] might be actually more useful. The ooDACE toolbox implements two instances of the Matérn

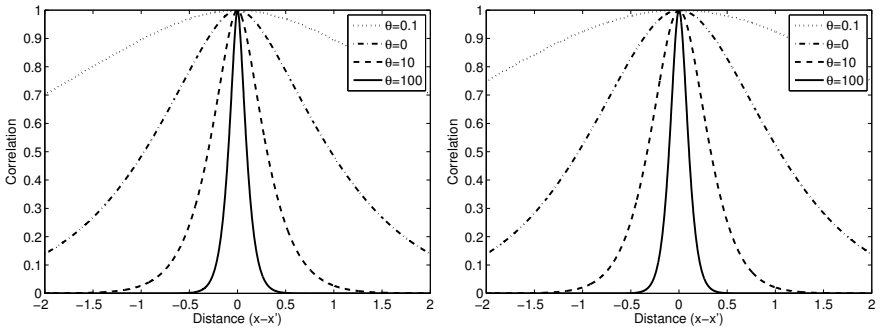


Figure 2.16.: The one-dimensional Matérn correlation functions with varying parameter θ: a) for $\nu = 3/2$ b) for $\nu = 5/2$.

correlation function, for $\nu = 3/2$ (*corrmatern32*) and $\nu = 5/2$ (*corrmatern52*), namely,

$$\psi(\mathbf{x}, \mathbf{x}')_{\nu=3/2} = \left(1 + \sqrt{3}l\right) \exp\left(-\sqrt{3}l\right),$$

$$\psi(\mathbf{x}, \mathbf{x}')_{\nu=5/2} = \left(1 + \sqrt{5}l + \frac{5l^2}{3}\right) \exp\left(-\sqrt{5}l\right),$$

with $l = \sqrt{\sum_{i=1}^{d} \theta_i (x_i - x_i')^2}$. The parameter ν of the Matérn correlation functions has a similar role as the p parameter, see Figure 2.16, a higher value is better suited for a rough behavior of the expensive function and vice versa.

2.2.2.7. Maximum Likelihood Estimation (MLE)

After choosing the general form of the correlation function for the problem at hand, its hyperparameters are identified using Maximum Likelihood Estimation (MLE). There are several variants of the likelihood that one can optimize, with the marginal likelihood the most widely used.

Marginal likelihood The natural log of the marginal likelihood is given by,

$$\ln(\mathcal{L}_{marginal}) = \frac{n}{2}\ln(2\pi) + \frac{n}{2}\ln(\sigma^2) + \frac{1}{2}\ln(|\Psi|) + \frac{1}{2\sigma^2}(\mathbf{y} - F\alpha)^T \Psi^{-1}(\mathbf{y} - F\alpha).$$

This can be simplified by taking the derivatives with respect to the α and σ^2, equaling it to zero and solving for α and σ^2. When we also remove

constant terms the (negative) concentrated Ln-likelihood is obtained as used in the ooDACE toolbox,

$$-\ln(\mathcal{L}_{marginal}) = -\frac{n}{2}\ln\sigma^2 - \frac{1}{2}\ln(|\Psi|) \tag{2.38}$$

where $\alpha = (F^T\Psi^{-1}F)^{-1}F^T\Psi^{-1}\mathbf{y}$ and $\sigma^2 = \frac{1}{n}(\mathbf{y} - F\alpha)^T\Psi^{-1}(\mathbf{y} - F\alpha)$, see also Section 2.2.2.1.

In essence, the first term, $-\frac{n}{2}\ln\sigma^2$, denotes the quality of the fit while the second term, $-\frac{1}{2}\ln(|\Psi|)$, represents a complexity penalty. Thus, the marginal likelihood automatically balances flexibility versus accuracy. However, the marginal likelihood depends on correct specification of the Kriging model for the data (e.g., choice of correlation function) and may not be robust enough when the Kriging model is misspecified.

Pseudo likelihood Leave-one-out cross validation is a popular method to assess the accuracy of a surrogate model, especially for interpolation based methods, and, hence, can also be used to tune the hyperparameters of Kriging. The Leave-One-Out (LOO) predictive Ln-probability [19, 13] is given by,

$$\ln(\mathcal{L}_{LOO}) = \sum_{i=1}^{n} -\frac{1}{2}\ln(\sigma_i^2) - \frac{(y^i - F\alpha - \mu^i)^2}{2\sigma_i^2} - \frac{1}{2}\ln(2\pi) \tag{2.39}$$

where $\mu^i = y^i - F\alpha - {}^{(\Psi^{-1}\mathbf{y})_i}/_{\Psi_{ii}^{-1}}$ and $\sigma_i^2 = {}^1/_{\Psi_{ii}^{-1}}$. In contrast to the marginal likelihood, the LOO predictive Ln-probability is independent of the model specification and, thus, may be more robust, though this has not been empirically validated.

2.2.3. Bibliography

[1] I. Couckuyt, K. Crombecq, D. Gorissen, and T. Dhaene. Automated response surface model generation with sequential design. In *First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering (CSC), Funchal, Portugal*, 2009.

[2] I. Couckuyt, F. Declercq, T. Dhaene, and H. Rogier. Surrogate-based infill optimization applied to electromagnetic problems. *Journal of RF and Microwave Computer-Aided Engineering: Advances in design optimization of microwave/rf circuits and systems*, 20(5):492–501, 2010.

[3] A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, Chichester, 2008.

[4] A.I.J. Forrester, A.J. Keane, and N.W. Bressloff. Design and analysis of "noisy" computer experiments. *AIAA Journal*, 44(10):2331–2336, 2006.

[5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Machine Learning*, 3:1157–1182, 2003.

[6] M. Hamada and C. F. J. Wu. Analysis of designed experiments with complex aliasing. *Quality Technology*, 24(3):130–137, 1992.

[7] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[8] V. R. Joseph. A bayesian approach to the design and analysis of fractionated experiments. *Technometrics*, 48(2):221–229, 2006.

[9] V. R. Joseph and J. D. Delaney. Functionally induced priors for the analysis of experiments. *Technometrics*, 49:1–11, 2007.

[10] V. R. Joseph, Y. Hung, and A. Sudjianto. Blind kriging: A new method for developing metamodels. *ASME Journal of Mechanical Design*, 130(3):031102–1–8, 2008.

[11] M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87:1–13, 2000.

[12] M.D. Morris, T.J. Mitchell, and D. Ylvisaker. Design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics*, 35(3):243–255, 1993.

[13] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[14] J. Sacks, W. J. Welch, T.J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, 4(4):409–435, 1989.

[15] T.J. Santner, B.J. Williams, and W.I. Notz. *The design and analysis of computer experiments*. Springer series in statistics. Springer-Verlag, New York, 2003.

[16] A.Y.D. Siem and D. den Hertog. Kriging models that are robust w.r.t. simulation errors. Technical report, Tilburg University, 2006.

[17] J. Staum. Better simulation metamodeling: The why, what, and how of stochastic kriging. In *Proceedings of the Winter Simulation Conference*, 2009.

[18] M.L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer-Verlag, 1999.

[19] S. Sundararajan and S. Sathiya Keerthi. Predictive approach for choosing hyperparameters in gaussian processes. *Neural Computation*, 13:1103–1118, 2001.

[20] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[21] J.C.F. Wu and M. Hamada. *Experiments: Planning, Analysis, and Parameter Design Optimization*. Wiley, New York, 2000.

2.3. Blind Kriging: Implementation and performance analysis

I. Couckuyt, A. Forrester, D. Gorissen, F. De Turck, T. Dhaene

Published in Advances in Engineering Software,

vol. 49, pp. 1-13, 2012.

Abstract

When analyzing data from computationally expensive simulation codes or process measurements, surrogate modeling methods are firmly established as facilitators for design space exploration, sensitivity analysis, visualization and optimization. Kriging is a popular surrogate modeling technique for data based on deterministic computer experiments. There exist several types of Kriging, mostly differing in the type of regression function used. Recently a promising new variable selection technique was proposed to identify a regression function in the Kriging framework. In this paper this type of Kriging, i.e., blind Kriging, has been efficiently implemented in Matlab® and has been extended. The implementation is validated and tested on several examples to illustrate the strength and weaknesses of this new, promising modeling technique. It is shown that the performance of blind Kriging is as good as, or better than ordinary Kriging. Though, blind Kriging comes at double the computational cost with respect to ordinary Kriging.

2.3.1. Introduction

Many complex real world phenomena are difficult to study directly using controlled experiments. Instead, the use of computer simulations has become commonplace as a feasible alternative. However, due to the computational cost of these high fidelity simulations, the use of surrogate modeling techniques has become indispensable. A popular surrogate model to approximate deterministic noise-free data is Kriging. First conceived by Danie Krige in geostatistics, these Gaussian process [5] based surrogate models are compact and cheap to evaluate, and have proven to be very useful for tasks such as optimization [8], design space exploration, visualization, prototyping, and sensitivity analysis [26].

A thorough mathematically treatment of Kriging is given by [19]. Basically, Kriging models fit data first on a regression function $f(\mathbf{x})$, and, subsequently, construct a stochastic process $Z(\mathbf{x})$ through the residuals.

$$Y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}) \tag{2.40}$$

By approaching the approximation problem from a Bayesian point of view, Kriging inherits a solid mathematical foundation with several useful properties, e.g., a closed formula for approximating the prediction variance.

Intuitively, the regression function in Kriging can be seen as the part trying to capture the general trend and thus the largest variations of the data, while the stochastic part takes care of small details and the interpolation of the data. However, choosing the right regression function for a set of data is a difficult and widely researched problem [14]. A simple approach would be just to apply a ranking method that assigns scores to the individual variables of the dataset. Afterwards, the most promising variables according to this score are selected to participate in a regression function. However, to also identify interactions between variables, features should be defined, i.e., an interaction between a set of variables (quadratic, linear-linear, etc.). The whole process of choosing the right terms is referred to as variable or feature selection. Note that the terms "*variable*" and "*feature*" are often used interchangeably, while in theory they have a different meaning [6].

Several feature selection methods are discussed in the literature. [10] proposed to apply a Bayesian feature selection method to Kriging. This form of Kriging is known as blind Kriging and is the main focus of this chapter.

A thorough analysis and performance study of blind Kriging is performed on a highly distinct set of mathematical and real life problems from the literature. In particular, blind Kriging is adapted to include a re-estimation of the hyperparameters during the feature selection process, as well as a normalization of the training dataset, which requires modified formulae. Furthermore, these changes are compared against traditional blind Kriging, as explained in [10], using several statistical methods, e.g., error on a dense test set, histogram plots, robustness analysis, etc.

First, a brief introduction of variable selection is given in Section 2.3.2. Then, Section 2.3.3 presents a practical overview of the blind Kriging. Subsequently, Section 2.3.4 describes the experimental setup used for testing this implementation against several applications. Sections 2.3.4.1-2.3.4.6 discuss the results of the various application benchmarks, illustrating strengths and weaknesses of blind Kriging. The last section concludes this paper and describes future work.

2.3.2. Feature selection

The time and space complexity of many surrogate modeling techniques (polynomials, radial basis functions, Kriging, etc.) scale exponentially with

[6] the term "*variables*" is used to denote the raw input variables. While, "*features*" denote artificial variables constructed from the raw input variables.

the dimensionality of the problem. In literature this is often noted as the "*curse of dimensionality*". By taking advantage of feature selection methods this can often be (somewhat) alleviated.

Feature selection is important if the data contains a large amount of variables. Usually, not all of these variables are relevant for the problem at hand, and several sets of variables might be correlated. Therefore, feature selection methods are applied to simplify the data. Irrelevant variables may be either left out completely (dimension reduction) or aggregated into variable subsets (known as a *features*). This results in an updated dataset, expressed in terms of these features, which is easier to analyze. The machine learning community is particularly active in the domain of feature selection methods as they often have to deal with enormous amounts of high dimensional data. Most research is focused on classification techniques, however, a lot of these techniques are also applicable to regression problems. It should be noted that the problem of selection the right features in regression is more commonly known as variable subset selection or subset selection in regression [14].

According to [12] feature selection methods can be classified into three categories: filters, wrappers and embedded methods.

Filters are pre-processing methods and, thus, not associated with a particular prediction model. Variable ranking is an important filter method and forms the foundation of many feature selection algorithms. Variable ranking methods assign a score to the features corresponding to their influence on the response according to some correlation criteria, e.g., Pearson's correlation. This allows the practitioner to select only those variables that are most sensitive to the response of interest. Note that this method does not consider the effects between variables. More advanced methods consider subsets of variables. These so-called variable subset selection mechanisms often use variable ranking as a principal component in their inner workings. Filter methods are said to be fast to compute and the resulting dataset is not particularly tuned for any prediction model and thus no bias is introduced, though, under certain assumptions optimality to a predictor can be proven.

Wrappers, popularized by [12], view the prediction model as a black box and only use them to assess the usefulness of a subset of variables. In general, algorithms in the wrapper category must define approximately four components: a search space of all possible variable subsets (1), a search strategy (2), a prediction model (3) and a performance measure of the prediction model to guide (and halt) the search (4). Two popular greedy search strategies are *forward selection* and *backwards elimination*, progressively incorporating features or discarding features respectively. Though, other strategies such as genetic algorithms may be used. While easy to use and applicable to any prediction model this approach requires a retraining of the prediction model several times.

The last category, *embedded methods*, tightly integrates with the prediction method (e.g., embedded in the training phase) and thus is often faster than

wrappers. An intuitive way to determine the influence of a feature is to calculate the derivative. This can be computed exactly for some models and training methods [21, 18]. Other prediction models, e.g., kernel based methods, inherently provide an approximation of the derivatives (sensitivity) [7]. A simple approach is to fit a polynomial on the data using least squares, including all candidates features. The coefficients of the polynomial represent the influence of a particular feature on the response and can be used to guide the selection of features in the final prediction model. An example of a kernel based method is Kriging. The Kriging model parameters quantify the variance of the response in each dimension. In fact, many kernel based methods, such as support vector machines [2], have this benefit as long as the appropriate kernel function is used. This approach, referred to as "Automatic Relevance Detection", basically attaches a weight to each dimension in the kernel function (the weights are determined using normal model parameter optimization routines). Depending on the kernel function the weights denote the influence, sensitivity, variance, etc. of the associated dimensions. Though this approach only works on the raw input variables and, hence, are only suited for dimension reduction (or adding dimensions). For more information see [6].

The feature selection mechanism used in blind Kriging belongs to the embedded class of methods, due to the tight coupling with Kriging. In particular, a Bayesian ranking method is used to rank several candidate features and a (greedy) forward selection procedure constructs a regression function. Subsequently, a Kriging model is used to measure the accuracy of the chosen regression function. Thus, the forward selection procedure is guided by a Bayesian ranking method and halted when the accuracy of the Kriging model increases again.

2.3.3. Fundamentals

Blind Kriging has been implemented in a stand alone Matlab® toolbox, the ooDACE toolbox[7], partly based on the original paper by [10] and associated R code. Pseudo-code of the complete blind Kriging algorithm is found in Algorithm 2.1.

As discussed in Section 2.2.2.3 on page 2-42, the estimation of $\hat{\beta}$ requires the correct parameters θ of the Kriging model to be available. Therefore, first, an ordinary Kriging model is built and this involves estimating θ. Further on, the identified θ parameters are kept fixed throughout the Bayesian forward selection procedure while appropriate regression terms are being selected. In the ideal case θ is optimized (fine-tuned) in each iteration once a new term is added. However, this adds a relative huge computational burden when constructing blind models, i.e., the computational cost is roughly

[7] The ooDACE Toolbox is freely available under an open source license (AGPLv3) for download at `http://sumo.intec.ugent.be/?q=ooDACE`

Algorithm 2.1 Pseudo-code of the blind Kriging model of the ooDACE toolbox.

```
1  X ← samples
2  b_1, ..., b_t  { Candidate features }
3  b = C^te  { Selected features }
4  θ_0 = max_θ likelihood(X, θ, b)
5  M_0 = construct(X, θ_0, b)  { Construct ordinary Kriging model }
6  α_0 = evaluateMeasure(M_0)  { Assess accuracy }
7  i = 0
8  while improvement(α)  { Accuracy improves? }
9     i = i + 1
10    β = rank(b_1, ..., b_t)
11    j = argmax_j(|β_j|)
12    b = b ∪ b_j
13    θ_i = max_θ likelihood(X, θ, b)  { Optional }
14    M_i = update(M_{i-1}, θ_i, b)  { Intermediate Kriging model }
15    α_i = evaluateMeasure(M_i)  { Assess accuracy }
16 end
17 θ_final = max_θ likelihood(X, θ, b)
18 M_final = update(M_i, θ_final, b)  { Final blind Kriging model }
```

equal to ordinary Kriging multiplied by the number of features considered. Hence, θ is only identified once before the Bayesian feature selection procedure starts, and a second time once the complete trend function has been chosen. Though, this still requires twice the computational cost compared to traditional Kriging.

The optimization strategy for the parameters θ of the Kriging model is of utmost importance as it has a large impact on the performance of (blind) Kriging. A nice overview of hyperparameter tuning strategies is given by [24], where a two-stage approach is suggested. First use a genetic algorithm (global search) to quickly explore the search space and obtain a rough estimate of θ. As Kriging is very sensitive to the last steps, while converging to the final accurate optimum, a hill climbing method (local search) is used to fine-tune the optimum. However, while such a two-stage process, i.e., a memetic algorithm [15], performs well with respect to finding the global optimum, it can be prohibitively expensive in higher dimensions. Therefore, another optimization strategy is used in the ooDACE toolbox which is outlined below.

The efficient calculation of the likelihood involves the factorization of the correlation matrix Ψ, and in this work a Cholesky decomposition is used requiring a time complexity of the order $O(n^3)$. To improve the efficiency, and to reduce the number of likelihood evaluations, derivative information

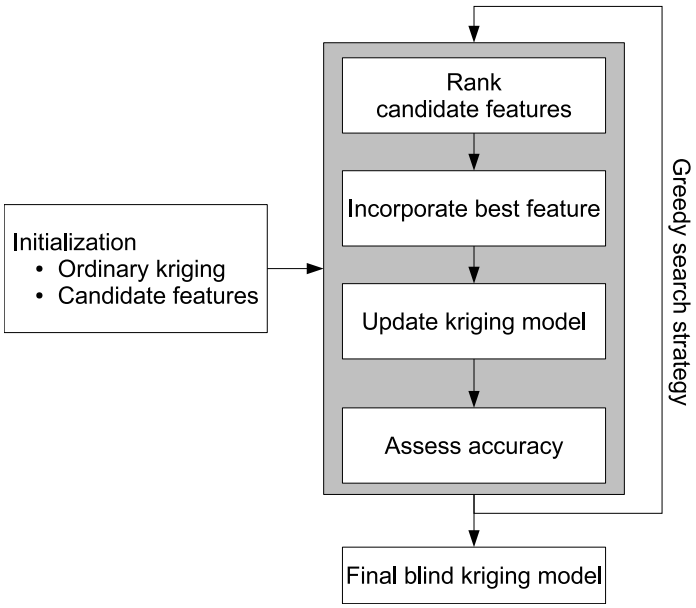


Figure 2.17.: Flow chart of the blind Kriging construction process.

is utilized in a Sequential Quadratic Programming (SQP) framework. The derivative of the concentrated likelihood can be calculated analytically or an adjoint [25] can be used. In both cases the SQP method is found to find competitive optima in comparison with the thorough search of a memetic algorithm while using significantly fewer likelihood evaluations. In addition, unlike a genetic algorithm, the used SQP method is deterministic resulting in a very robust blind Kriging implementation. In this work, the derivatives of the likelihood function are calculated analytically instead of using the adjoint method since the difference in computation time is negligible for the problems in this paper.

The basic algorithm is depicted in Figure 2.17. After choosing the initial set of candidate features and constructing the initial Kriging model, the candidate features are ranked using the Bayesian variable selection method. Subsequently, a search strategy selects a new promising feature (according to the ranking) which is incorporated in the ordinary Kriging model. This intermediate Kriging model is scored against a measure and the features are ranked again. This process is repeated until the accuracy of the Kriging model stops improving. Several other adjustments were made to this basic algorithm to improve efficiency. These improvements are discussed in the remainder of this section.

Arguably the most important factor when determining relevant variable interactions is the method used to guide the search through the feature

space, and, closely related, the criterion to stop adding terms. The leave-one-out cross validation has been used here, combined with a heuristic stopping criterion, as described in [10]. In addition, the ooDACE toolbox also supports the use of an unbiased hold-out set to validate the intermediate blind Kriging models as well as the likelihood itself, though this functionality has not been used in this paper.

Recall that Kriging requires inverting the correlation matrix Ψ. Depending on the number and the distribution of samples the correlation matrix may become close to singular, resulting in inaccuracies. Analogous to the DACE toolbox [13] and as suggested by [3] the ooDACE toolbox uses Cholesky and QR decompositions to efficiently compute the BLUP and likelihood. In addition, the likelihood score is set to infinity when a bad condition number is encountered, and so unstable Kriging models are effectively avoided.

The scaling of the sample data is also an important issue. In [10] all sample data used to be in the interval $[1, 3]$ (three-level factorial design). In the ooDACE toolbox, as in the DACE toolbox, all data is normalized (into a standardized dataset), i.e., it scales the input and output data such that the data is distributed with mean 0 and variance 1,

$$\mu(\mathbf{x}) = 0 \; and \; \sigma(\mathbf{x}) = 1, \tag{2.41}$$

$$\mu(\mathbf{y}) = 0 \; and \; \sigma(\mathbf{y}) = 1, \tag{2.42}$$

this scaling is based on the hypothesis that, although being a distribution-free spatial interpolator, Kriging achieves its maximal efficiency only when the training data follows a Gaussian distribution. Hence, samples and values should be transformed to the *"Gaussian domain"* [16] when fitting, and afterwards, the BLUP, the uncertainty measures and all other estimates are transformed back to the original domain. While, the normalization of the data as described above is far from being an exact conversion to the *"Gaussian domain"*, it does help to reduce the effects of outliers, i.e., extreme values in the input as well as output domain, and makes the scaled data easier to model. However, this scaling requires changing the orthogonal polynomial coding and R matrix equations. For arbitrarily bounds on the data, equations 2.25-2.26 (page 2-44) can be written as,

$$x_{j,l} = \frac{\sqrt{3}}{\sqrt{2}} \left(\frac{\mathbf{x_j}}{\mathbf{l}_3} \right), \tag{2.43}$$

$$x_{j,q} = \frac{1}{\sqrt{2}} (3 \left(\frac{\mathbf{x_j}}{\mathbf{l}_3} \right)^2 - 2), \tag{2.44}$$

and equations 2.27-2.28 (page 2-45) as,

$$\mathbf{r}_l = \frac{3 - 3\psi(\mathbf{l}_3 - \mathbf{l}_1)}{3 + 4\psi(\mathbf{l}_2 - \mathbf{l}_1) + 2\psi(\mathbf{l}_3 - \mathbf{l}_1)}, \tag{2.45}$$

$$\mathbf{r}_q = \frac{3 - 4\psi(\mathbf{l}_2 - \mathbf{l}_1) + \psi(\mathbf{l}_3 - \mathbf{l}_1)}{3 + 4\psi(\mathbf{l}_2 - \mathbf{l}_1) + 2\psi(\mathbf{l}_3 - \mathbf{l}_1)}, \tag{2.46}$$

where $\mathbf{l}_i$ denotes the i^{th} level in a factorial design. These can be defined as,

$$\mathbf{l}_1 = min(X), \tag{2.47}$$

$$\mathbf{l}_3 = max(X), \tag{2.48}$$

$$\mathbf{l}_2 = mean(X), \tag{2.49}$$

min and *max* take the columnwise minimum and maximum, respectively, of the sample matrix X such that $\mathbf{l}_1$ and $\mathbf{l}_3$ effectively provide the tightest bounding box of the data. $\mathbf{l}_2$ is the columnwise mean of the sample matrix X, when using the aforementioned scaling this is equal to $\frac{\mathbf{l}_1+\mathbf{l}_3}{2} \cong \mathbf{0}$.

Currently, the ooDACE toolbox only supports linear and quadratic trend functions. In fact, quoting from [20],

> "...for most well-defined physical system, only relatively low-order correlations of the input variables are expected to have a significant impact upon the output, and high-order correlated behavior of the input variables is expected to be weak".

Thus, it is not crucial to consider higher interactions than quadratic. This is noticed by other authors such as [11].

Identifying higher order interactions (cubic and higher) is only possible when a greater number of levels is considered in the factorial design. Moreover, the R matrix and $\mathbf{r_l}$,$\mathbf{r_q}$ equations have to be adapted accordingly and appropriate coding and $\mathbf{r}$ equations should be added. In addition, the R matrix will grow rapidly in size when more candidate features are considered.

Finally, the ooDACE toolbox is limited to the Gaussian correlation function and the exponential correlation function, though, other correlation functions could easily be added as long as they can be written in the required product correlation structure notation (see Equation 2.23 on page 2-43). For the Gaussian and exponential correlation function the formulae are given by

$$\psi_j(d_j) = e^{-\theta_j \cdot d_j^2}, \tag{2.50}$$

and,

$$\psi_j(d_j) = e^{-\theta_j \cdot |d_j|}, \tag{2.51}$$

respectively.

2.3.4. Performance

Intuitively blind Kriging can be expected to do better on problems where the regression function is able to approximate the general trend of the data. For instance, when using linear and quadratic effects as candidate variables then problems that (strongly) exhibit these effects would do rather well of course. Naturally, the behavior of the response is unknown and thus no regression function can be defined a priori.

The performance of the ooDACE toolbox is applied to a very distinct set of real-life problems. First, the performance of this blind Kriging implementation is validated on three examples taken from the original blind Kriging paper [10], namely, a sealing experiment (Section 2.3.4.1), piston slap (Section 2.3.4.2) and the borehole model (Section 2.3.4.3). Whenever a comparison is made to the findings of Joseph et al. this will be referred to as *the reference paper*. Afterwards, blind Kriging is applied to a novel problem from hydrology, quantifying the effects of pesticide leaching on ground water in Europe. Thereafter, blind Kriging is applied to a problem from Mechanical Engineering which has no clear trend, i.e., Section 2.3.4.5, the application data has been obtained from [4]. Subsequently, blind Kriging is tested on a two dimensional mathematical function, i.e., the Branin function. The results are found in the following sections.

All tests were performed using the standard Gaussian correlation function. For several examples, the implementation introduced in this paper is also compared against the freely available DACE toolbox [13]. The DACE toolbox is configured as ordinary Kriging, i.e., a constant regression function (*@regpoly0*), and the correlation function is set to Gaussian (*@corrgauss*). The ordinary Kriging model produced by the blind Kriging code is also taken into account.

2.3.4.1. Sealing experiment

The first dataset was directly obtained from [10]. Therefore, a full description of the problem can be found in there. In summary, the data consists of eight input parameters and one output parameter resembling the gap lift in an engine block and head sealing experiment. There are 27 observations which form an orthogonal array. The goal of this experiment is to validate the correctness of the implementation and compare against the original findings of Joseph et al.

First the DACE toolbox was used to construct an ordinary Kriging model of this dataset. The obtained ordinary Kriging model serves as a base for the Bayesian forward selection procedure to produce the final blind Kriging model. The evolution of the cross validated prediction error (*cvpe*; leave-one-out) versus the chosen terms is shown in Figure 2.18. Starting from a rather high leave-one-out score (in comparison to the reference paper) for the ordinary Kriging model (OK), the score decreases and settles at a value

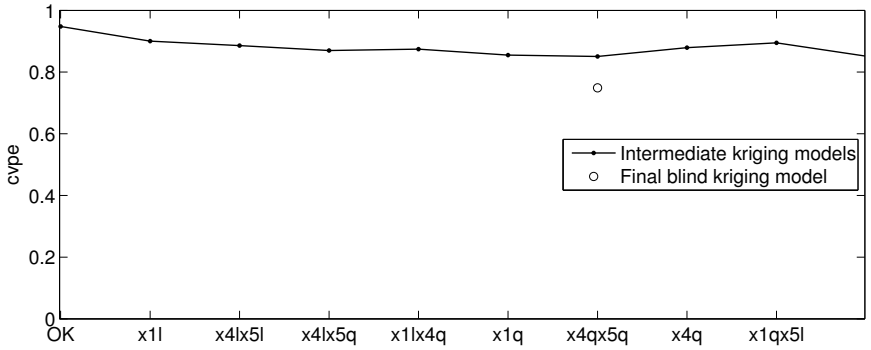
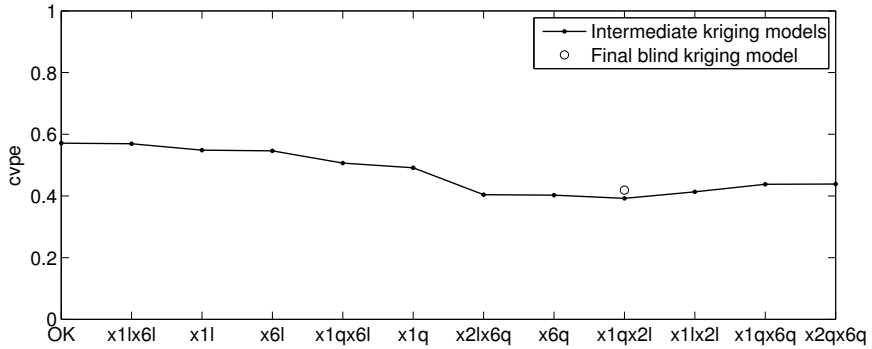


Figure 2.18.: Evolution of the Bayesian feature selection phase (sealing experiment). Using the DACE toolbox for the initial Kriging model (without hyperparameters re-estimation).

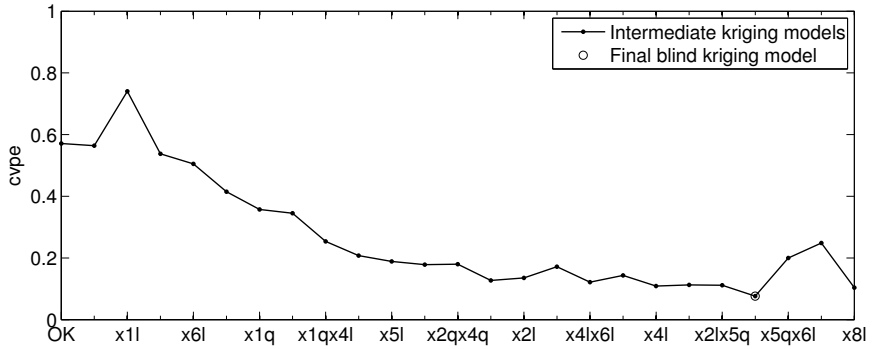
of approximately 0.87, after choosing six extra terms. After re-estimating the θ parameters, this time using the optimization strategy discussed in Section 2.3.3, there is much improvement to be found (the circle beneath the curve). Although, curiously, these results are much worse in comparison to the reference paper. Looking at the chosen features, see Equation 2.52, they do not correspond to the regression function found by Joseph et al.

$$1 + x_{1,l} + x_{4,l} \cdot x_{5,l} + x_{4,l} \cdot x_{5,q} + x_{1,l} \cdot x_{4,q} + x_{1,q} + x_{4,q} \cdot x_{5,q} \tag{2.52}$$

Considering the poor score of the first constructed ordinary Kriging model it was decided to use the optimization strategy of the ooDACE toolbox to identify the θ parameters of the ordinary Kriging model. Results are depicted in Figure 2.19a. As can be seen a slightly better initial leave-one-out score is calculated and using this set of θ parameters the feature selection procedure is able to select the right features, reducing the score to 0.39 after eight terms. The final regression function of the blind Kriging model (without the coefficients) is given by (2.53). This regression function contains the same terms as the reference paper, though selected in a slightly different order. The difference is found in the last two terms $x_{6,q}$ and $x_{1,q} \cdot x_{2,l}$, which give lead to a rise in cross validation score in the reference paper and thus are not selected. Here it is found that they slightly lower the score, explained by a different set of θ parameters. Re-estimating the θ parameters it can be seen that the cross validation score for the blind Kriging model is actually worse with the newly identified parameters. After some testing it is found that those two last terms were not exactly appropriate. If those terms are left out of the regression function the final cross validation is in the same range as the reference paper.



(a) The ooDACE toolbox. Remark that the re-estimated model parameters of the final blind Kriging model result in a worse cvpe score, meaning the optimized likelihood score and the cvpe score conflict in this case (without hyperparameters re-estimation).



(b) The ooDACE toolbox. Note that not all chosen features are shown on the x-axis to avoid cluttering (with hyperparameters re-estimation; the cvpe scores are *after* the hyperparameters re-estimation).

Figure 2.19.: Evolution of the Bayesian feature selection phase (sealing experiment).

$$1 + x_{1,l} \cdot x_{6,l} + x_{1,l} + x_{6,l} + x_{1,q} \cdot x_{6,l} \\ + x_{1,q} + x_{2,l} \cdot x_{6,q} + x_{6,q} + x_{1,q} \cdot x_{2,l} \tag{2.53}$$

This shows that the calibration of the initial ordinary Kriging model greatly influences the Bayesian feature selection procedure. Moreover, the second approach clearly shows that the implementation of blind Kriging is competitive with the reference paper. Any difference between the two is most likely due to a different set of θ parameters. In addition, note that the original dataset takes on exactly three levels (1, 2 and 3). Whereas in the proposed implementation the models are fitted on normalized data (see Section 2.3.3).

The behavior described above is mostly due to the optimized θ parameters not matching the current regression function anymore. In essence, those θ parameters are describing the function that is the difference between the original observations *minus* the current regression function. As optimization is not that expensive when using derivative information and for testing purposes it is decided to re-estimate the θ parameters after every added term. Results using this approach are shown in Figure 2.19b. Many more terms are included in the regression function resulting in a cross validation score as low as 0.1.

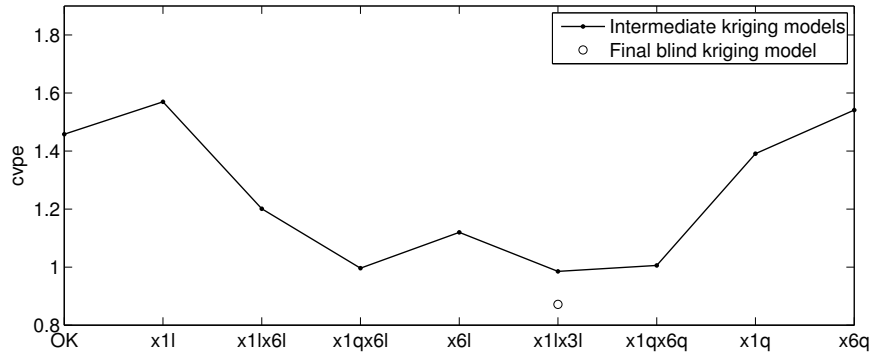
2.3.4.2. Piston slap

The second experiment is a dataset describing engine noise due to piston secondary motion (piston slap). For more information the reader is again referred to [10]. The dataset consists of six inputs and one output (the noise). This time, the dataset is directly approximated using solely the blind Kriging implementation and leave-one-out cross validation to guide the feature selection. The evolution of the feature selection stage can be found in Figure 2.20a. The resulting regression function is very similar to the reference paper, except for the last two terms.

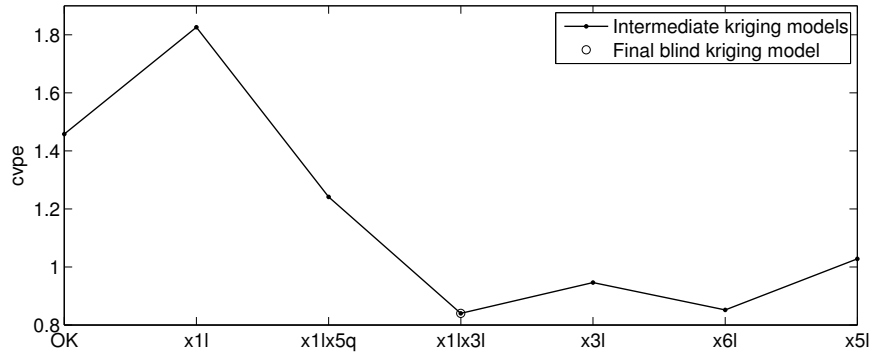
$$1 + x_{1,l} + x_{1,l} \cdot x_{6,l} + x_{1,q} \cdot x_{6,l} + x_{6,l} + x_{1,l} \cdot x_{3,l} \tag{2.54}$$

At the fourth step the term $x_{6,l}$ is chosen, giving an increase in the cross validation score. However, by adding $x_{1,l} \cdot x_{3,l}$ the score decreases again, settling at an even lower value than before. While $x_{1,l} \cdot x_{3,l}$ may be a good feature, $x_{6,l}$ is clearly not. However, by using a greedy forward selection strategy previously selected terms are never reconsidered. It may be worthwhile to adapt the search strategy to eliminate (or skip) terms that result in a (temporary) increase of the score, though, one should of course not violate the principles of effect hierarchy and heredity.

Comparing these results with the paper of Joseph et al. it is immediately clear that the cross validation score is substantially lower for the proposed



(a) The ooDACE toolbox (without hyperparameters re-estimation).



(b) The ooDACE toolbox (with hyperparameters re-estimation; the cvpe scores are *after* the hyperparameters re-estimation).

Figure 2.20.: Evolution of the Bayesian feature selection phase (piston slap).

Model type	27 samples		200 samples	
	AEE	Improvement	AEE	Improvement
Blind Kriging	6.31	66%	0.92	84%
Ordinary Kriging	9.33	51%	2.24	61%
DACE toolbox	19.62	baseline	5.73	baseline

Table 2.2.: Accuracy of the approximations on a test set (borehole model).

blind Kriging implementation. Here, two more terms are included in the regression function causing a lower cross validation score.

As in the previous example the configuration is slightly changed so as to re-estimate the hyperparameters after every added term. The surprising results are depicted in Figure 2.20b. The only common feature with the no hyperparameters re-estimation case (aside from the mean) is $x_{1,l}$ and only three extra terms are identified. Nevertheless, with this limited set of features blind Kriging is still able to achieve a cross validation score approximately the same as without re-estimating the hyperparameters.

2.3.4.3. Borehole model

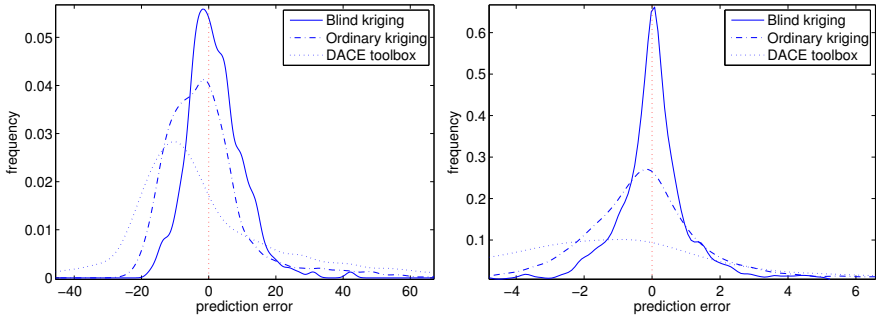
The final validation experiment is a simple analytical model to calculate the flow rate through a borehole [27]. There are eight inputs and the process is described by the following functions,

$$rr_w = \ln(\frac{r}{r_w}), \tag{2.55}$$

$$flow\,rate = \frac{2\pi T_u(H_u - H_l)}{rr_w[1 + (\frac{2LT_u}{rr_w r_w^2 K_w} + \frac{T_u}{T_l}]}, \tag{2.56}$$

The same orthogonal array of 27 samples of Section 2.3.4.1 is used to create a dataset of the borehole model. In addition, an optimized maximin Latin Hypercube Design (LHD; [9]) of 200 samples is constructed. These two datasets are used to create an ordinary Kriging model and blind Kriging model (without hyperparameters re-estimation) using the code presented in this paper. In addition, another ordinary Kriging produced by the DACE toolbox is taken into account. Hence, in total there are three Kriging models of which an error is calculated on a separate test set of $k = 6561$ samples. The Average Euclidean Error (AEE) is used, which is defined by,

$$AEE(\mathbf{a}, \mathbf{b}) = \frac{1}{k}\sum_{i=1}^{k}\sqrt{(a_i - b_i)^2}. \tag{2.57}$$



(a) 27 samples. **(b)** 200 samples.

Figure 2.21.: Density plot for the prediction errors (borehole model).

Results are found in Table 2.2. In both cases ordinary Kriging achieves a great improvement over the DACE toolbox due to a better optimization strategy. Furthermore, blind Kriging offers for this problem a vast improvement over ordinary Kriging itself and, hence, the DACE toolbox. In addition, histogram plots, see Figure 2.21, of the prediction errors on the test set also compare in favor of blind Kriging. The resulting regression functions (in the order they are selected) of blind Kriging for the case of 27 (Equation 2.58) and 200 samples (Equation 2.59), respectively, are,

$$\begin{aligned} &1 + x_{1,l} + x_{4,l} + x_{7,l} + x_{1,q} + x_{6,l} \\ &+x_{1,l} \cdot x_{4,l} + x_{1,l} \cdot x_{7,l} + x_{1,l} \cdot x_{6,l} + x_{8,l} \end{aligned} \tag{2.58}$$

$$\begin{aligned} &1 + x_{1,l} + x_{7,l} + x_{4,l} + x_{6,l} + x_{1,q} + x_{8,l} + x_{1,l} \cdot x_{4,l} \\ &+x_{1,l} \cdot x_{7,l} + x_{1,l} \cdot x_{6,l} + x_{1,l} \cdot x_{8,l} + x_{4,l} \cdot x_{7,l} \\ &+x_{6,l} \cdot x_{7,l} + x_{7,q} + x_{1,q} \cdot x_{4,l} + x_{4,l} \cdot x_{8,l} + x_{6,l} \cdot x_{8,l} \\ &+x_{2,l} + x_{7,l} \cdot x_{8,l} + x_{1,l} \cdot x_{7,q} + x_{1,q} \cdot x_{7,l}. \end{aligned} \tag{2.59}$$

It is obvious that the two regression functions are quite similar aside from the order in which the features were chosen. In fact, the regression function of the second case is a superset of the one from the first case. In particular, 11 extra terms are identified likely due to more information being available (more samples). While Joseph et al. reported only a linear effect for x_1 as regression function, it is not immediately clear whether this term was chosen manually or using a search strategy as we have in this paper.

In summary, from the experiments of Sections 2.3.4.1, 2.3.4.2 and 2.3.4.3 it is clear that the implementation produces similar results as Joseph et al. [10].

Model type	AEE	Improvement
Blind Kriging	0.92	51%
Ordinary Kriging	1.94	−3%
DACE toolbox	1.88	baseline

Table 2.3.: Accuracy of the approximations on a test set (EuroPEARL model).

2.3.4.4. EuroPEARL

This test case consists of a dataset of 51319 samples generated by the EuroPEARL model [17, 23]. EuroPEARL models the leaching of pesticides, taking into account transient flown, hydrodynamic dispersion, nonlinear adsorption, degradation, and uptake of pesticides by plant roots. The model is developed at the European scale mainly due ground water being a major drinking source for Europe. In particular EuroPEARL consists of a link between the one-dimensional, multi-layer, mechanistic pesticide leaching model PEARL and a Geographical Information System (GIS). More details about EuroPEARL and the associated metamodeling efforts, denoted MetaPEARL, is found in [22].

The disadvantage of such a process-based pesticide-leaching model are the rather large number of parameters of which some are significantly less relevant than others. Hence, the use of feature selection techniques may prove to be indispensable for this problem. The EuroPEARL configuration for this paper has six input parameters describing the response.

For this test case, a training dataset of 100 samples arranged in a LHD is subsampled from the full dataset. The remaining samples of the dataset are taken as a test set to calculate the true error, using the AEE function. The same approximation models as the previous section are considered, namely, blind Kriging, ordinary Kriging and the DACE toolbox. Note, in contrast to the previous section, the blind Kriging model is configured to re-estimate the hyperparameters after every added term.

The true errors of the constructed Kriging models are found in Table 2.3. While ordinary Kriging has a slightly worse accuracy than the DACE toolbox, the blind Kriging approximation is more than twice as accurate. Looking at the histogram plots, Figure 2.22, it is seen that ordinary Kriging and the DACE toolbox are biased towards the right, namely, predicting smaller values than the real response. Using only 100 samples, blind Kriging is able to select a trend function that follows the behavior of the whole test set.

EuroPEARL is the perfect example for blind Kriging, a very small dataset which represents the features of the full response quite well. Equation 2.60 represents the final identified regression function of blind Kriging.

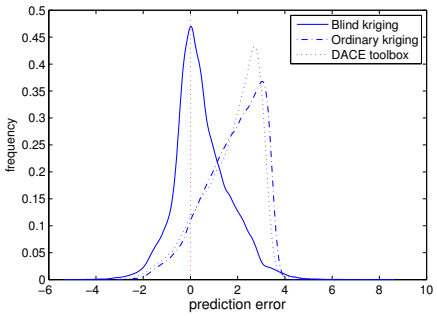


Figure 2.22.: Histogram plot (EuroPEARL model).

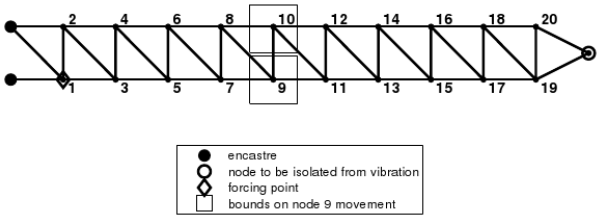


Figure 2.23.: Truss structure.

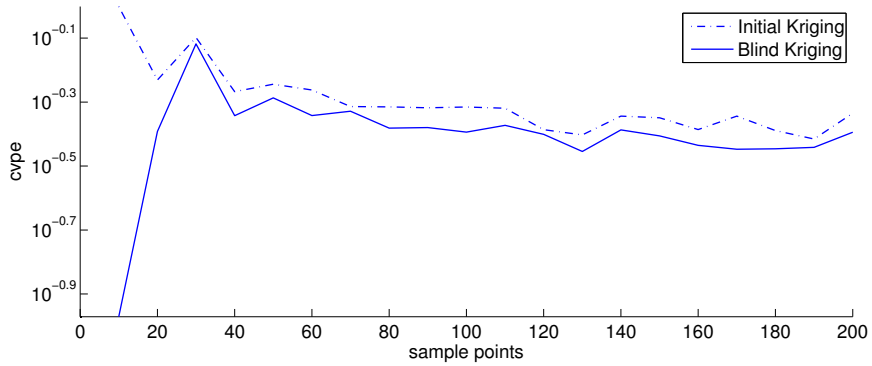
$$1 + x_{1,l} \cdot x_{5,q} + x_{3,l} \cdot x_{5,q} + x_{1,q} \cdot x_{3,q} + x_{4,q} \cdot x_{5,q} + x_{3,q} \cdot x_{5,l} \tag{2.60}$$

2.3.4.5. Truss dataset

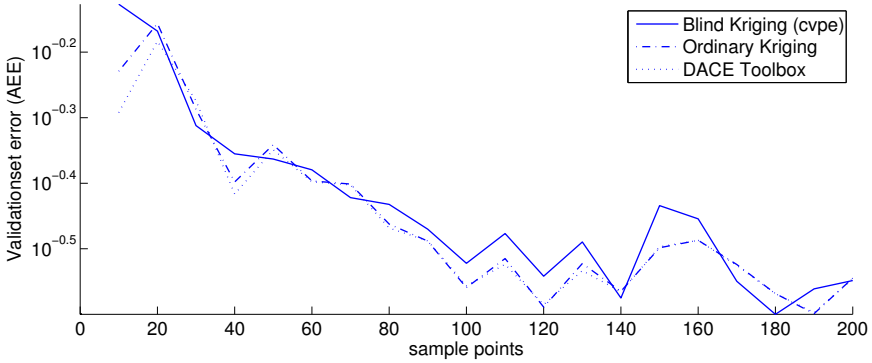
The fifth dataset describes a structural dynamics problem. The problem is the optimal design of a two-dimensional truss, constructed by 42 Euler-Bernoulli beams, see Figure 2.23. The goal is to identify a design that is optimal (or close to) with respect to passive vibration isolation. To that end, a force is induced on a base node of the structure and the force attenuated through the structure is measured on the tip of the structure. A full description is given in [4].

There are four input parameters defining the position of nodes nine and ten in the structure and one output parameter, i.e., the stress that the outermost node (the tip) receives. Note that the truss dataset is not used for optimization purposes, instead the goal is the reproduce the landscape as accurate as possible. To that end, 20 datasets are constructed each arranged in an optimized maximin LHD. The datasets only differ in the number of observations, which ranges from 10 to 200 samples with steps of 10 samples.

For each dataset, a DACE toolbox model and a blind Kriging model (with hyperparameters re-estimation) have been constructed. In addition,



(a) Evolution of the leave-one-out cross validation error.

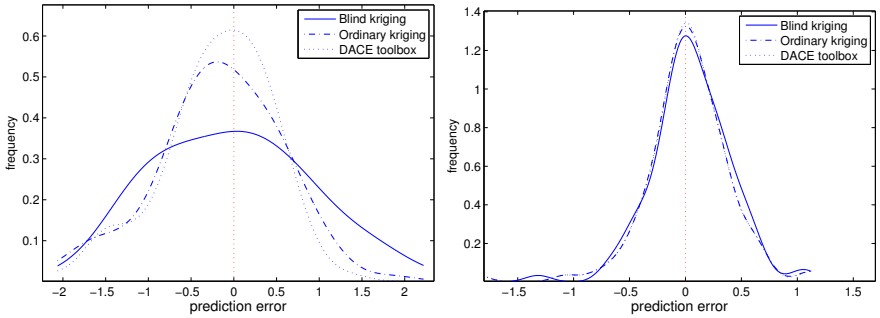


(b) Evolution of the AEE versus the number of samples.

Figure 2.24.: Accuracy of the prediction models (truss).

the ordinary Kriging model resulting from the blind Kriging construction process is included as well. An estimation of the true error is obtained for each model on a separate random test set of $k = 100$ samples. Results are shown in Figure 2.24. While, the evolution of the leave-one-out score promises increased accuracy (Figure 2.24a), little of this can be seen in the final AEE scores (Figure 2.24b). Looking at this second plot it is seen that adding terms rarely increases the accuracy. This is explained by the fact that the truss datasets have no clearly defined trend. At least, no obvious linear or quadratic effects can be seen. Perhaps if the set of candidate variables are extended with more complex interactions terms a good regression function can be found. However, this requires some alterations in the existing implementation and thus is considered future work.

Density plots of the prediction errors for the case of 10 and 140 samples are shown in Figure 2.25. Not much improvement is found, on the contrary, the histogram for the case of 10 samples is hardly noticeable in favor of



(a) 10 samples. **(b)** 140 samples.

Figure 2.25.: Density plot for the prediction errors (truss).

ordinary Kriging. Despite its attractiveness, blind Kriging is not suited for every problem available. Care should be taken when choosing the candidate variables. Ideally a domain expert is able to select plausible interactions for the given problem.

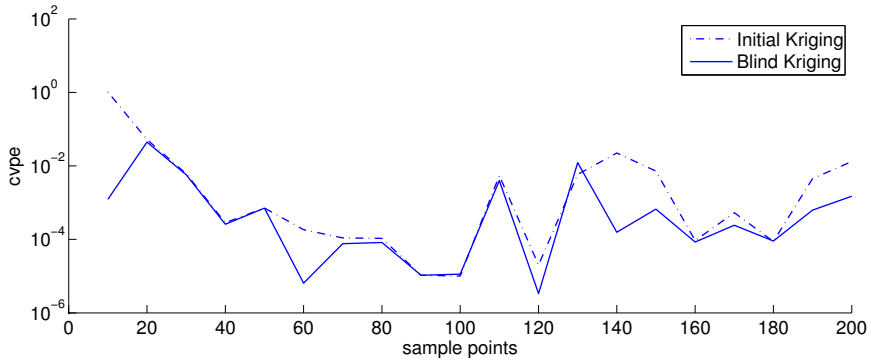
2.3.4.6. Branin function

The Branin function is a well-known benchmark function for optimization. The Branin function is defined by Equation 2.61. Again, 20 datasets arranged in an optimal maximin LHD [1] were used to construct the different Kriging models. Regard that the Branin function is not used here for optimization purposes but the intent is to reproduce the landscape of the Branin function as accurately as possible.

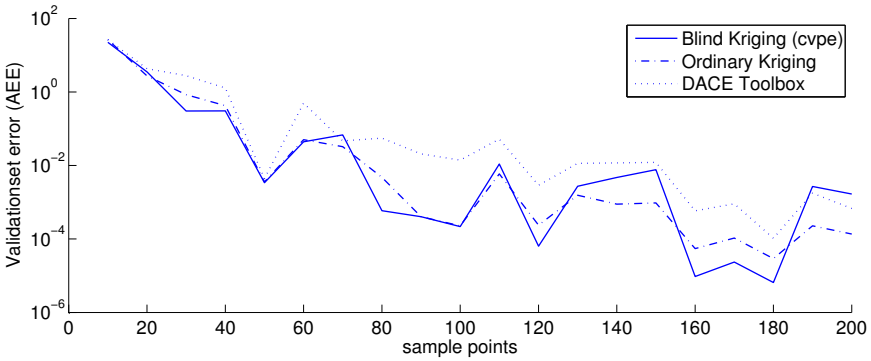
$$f(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10 \quad (2.61)$$

Figure 2.26a depicts the final cross validated leave-one-out score (*cvpe*) of the blind Kriging model (with hyperparameters re-estimation) for each dataset, while the initial baseline score of the ordinary Kriging model is included as well. As expected the *cvpe* of blind Kriging is always lower than ordinary Kriging. An estimation of the true error is obtained for each model on a separate and very dense dataset arranged in an uniform grid of $k = 2500$ samples. The AEE errors on this dataset versus the number of samples are shown in Figure 2.26b.

The performance differences between the blind Kriging and ordinary Kriging models are quite small. There is no obvious better Kriging model for this problem which can be explained by the abundance of data on this low dimensional problem, i.e., the stochastic part is always able to capture most of the variance making the regression function less important. Hence,

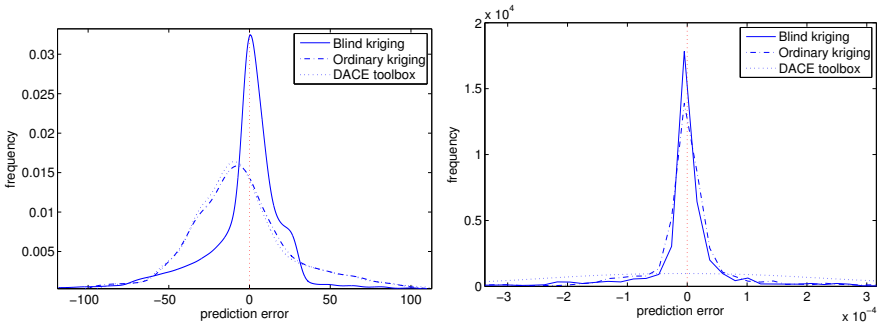


(a) Evolution of the leave-one-out cross validation error.



(b) Evolution of the AEE versus the number of samples.

Figure 2.26.: Accuracy of the prediction models (Branin 1).



(a) Dataset of 10 samples. **(b)** Dataset of 140 samples.

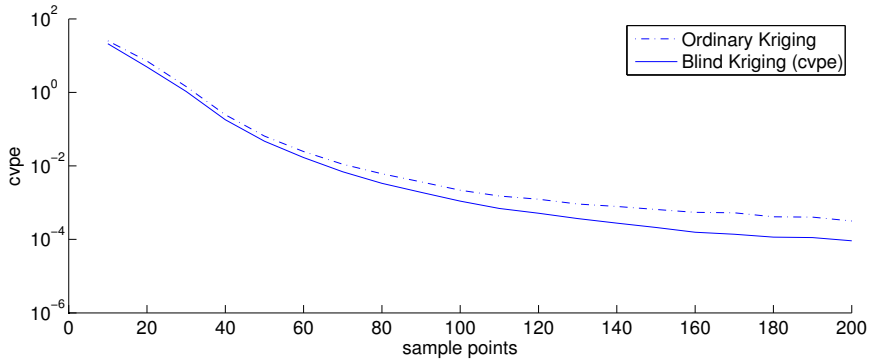
Figure 2.27.: Density plot for the prediction errors (Branin 1).

the differences on the AEE scores are insignificant. More of interest are the distributions of the prediction errors. Therefore, two density plots of the prediction errors for the datasets of 10 and 140 samples are depicted in Figure 2.27.

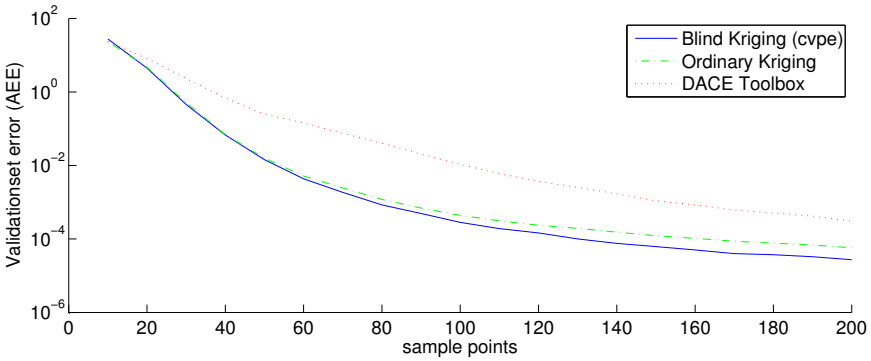
Surprisingly, looking at Figure 2.27a the blind Kriging model has a much better distribution of the errors than the other two models, even though it has a slightly higher AEE score. It is clear that using a suitable regression function removes much bias and provides a more stable foundation for the stochastic process of Kriging to build on. However, as the number of samples increase blind Kriging has a harder time distinguishing itself from the classical Kriging models (see Figure 2.27b). While considering the simplicity of the problem, i.e., only 2 dimensions, it is straightforward that any type of Kriging model has little trouble approximating the landscape well. Easily reaching an accuracy score of 10^{-4}.

To have an idea of the robustness of blind Kriging with respect to the sample distribution, the Branin function test is repeated using random designs instead of an optimal LHD. For each number of samples, ranging from 10 to 200 samples in steps of 10 samples, 1000 uniform random designs are constructed. Thus, the fitting of the Kriging models under consideration is repeated 1000 times for each sample size.

Similar plots as above are possible, including error bars. Though, the error bars make the evolution plot somewhat cluttered, as such only the mean accuracy is plotted in Figure 2.28. Note that the evolution is much smoother than the previous evolution plots using a LHD. This time, blind Kriging performs consistently better than the other approximations on the true error, though not by much. It is obvious that blind Kriging is still somewhat sensitive to the distribution of the samples. Logically, correct feature selection is only possible as long as the dataset is a decent representation of the general behavior of the simulation code.

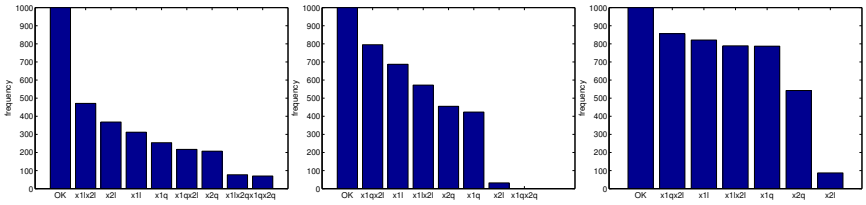


(a) Evolution of the average leave-one-out cross validation error.



(b) Evolution of the average AEE versus the number of samples.

Figure 2.28.: Average accuracy of the prediction models (Branin 2).



(a) 10 samples. **(b)** 100 samples. **(c)** 200 samples.

Figure 2.29.: Histogram plot of the chosen features (Branin 2).

More interesting is Figure 2.29 which depicts three histogram snapshots (10, 100 and 200 samples) describing the frequency each feature has been chosen (out of 1000). The leftmost bar (OK) is the mean which is always included in the regression function. As expected, the more samples are available the more accurate the selection of terms is, more specific, less terms are selected wrongfully and correct terms are selected more often.

2.3.5. Conclusion

This paper discussed blind Kriging, the associated Bayesian forward selection method and how it compares to traditional Kriging. An efficient implementation of blind Kriging, with numerous additions such as hyper-parameters re-estimation and normalization of the training data, has been tested and validated on six different problems to illustrate strengths and weaknesses. Blind Kriging is able to identify good regression functions for problems that have a clear trend. In particular, the training data should cover non-linearities in the domain for blind Kriging to work nicely. On the other hand, if the considered features do not contain the interactions exhibited by the data then blind Kriging is not able to identify a good regression function and does not improve on ordinary Kriging. In some cases blind Kriging can be misguided by deceptive data and select wrong features, effectively decreasing the accuracy of the final model. In that regard more work is to be done on the search strategy to select candidate features and the associated metric to guide this search. Finally, it should be noted that when enough data is available to cover the domain ordinary Kriging performs equally well as blind Kriging and even slightly outperforms blind Kriging for some datasets.

In summary, blind Kriging is a valuable tool to approximate sparse data obtained from expensive simulation codes. Though, the resulting blind Kriging model should be carefully analyzed, if possible using an independent test set. In particular, blind Kriging is more interesting for difficult, high dimensional, problems where limited data is available.

Future work includes investigating the impact of 4-level and higher

factorial designs to identify higher order interactions, researching other optimization strategies for choosing the best subset of variables (this includes performance metrics to guide the optimization), etc.

2.3.6. Bibliography

[1] E.R. Dam, B.G.M. van Husslage, D. den Hertog, and J.B.M. Melissen. Maximin Latin hypercube designs in two dimensions. *Operations Research*, 55(1):158–169, 2007.

[2] Jos de Brabanter. *LS-SVM Regression Modelling and its Applications*. PhD thesis, Katholieke Universiteit Leuven, 2004.

[3] A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, Chichester, 2008.

[4] A.I.J. Forrester and D.R. Jones. Global optimization of deceptive functions with sparse sampling. In *Proceedings of the AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 15. AIAA, 2008.

[5] Marks Gibbs and David J. C. Mackay. Efficient implementation of gaussian processes. Technical report, Department of Physics, Cavendish Laboratory, Cambridge University, 1997.

[6] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Machine Learning*, 3:1157–1182, 2003.

[7] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[8] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[9] V. R. Joseph and Y. Hung. Orthogonal-Maximin Latin hypercube designs. *Statistica Sinica*, 18:171–186, 2008.

[10] V. R. Joseph, Y. Hung, and A. Sudjianto. Blind kriging: A new method for developing metamodels. *ASME Journal of Mechanical Design*, 130(3):031102–1–8, 2008.

[11] Andy J. Keane and Prasanth B. Nair. *Computational approaches for Aerospace Design, The Pursuit of Excellence*. John Wiley & Sons Ltd., The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2005.

[12] R. Kohavi and G. John. Wrappers for feature selection. *Artificial Intelligence*, 97:273–324, 1997.

[13] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.

[14] A.J. Miller. *Subset Selection in Regression*. London: Chapman & Hall, 1990.

[15] P. Moscato and C. Cotta. Memetic algorithms. In *Optimization Techniques in Engineering*, pages 53–85. Springer-Verlag, 2004.

[16] Eulogio Pardos-Igúzquiza and Peter Dowd. Empirical maximum likelihood kriging: The general case. *Mathematical Geology*, 37(5):477–492, 2005.

[17] J.D. Pineros-Garcet, M. Vanclooster, A. Tiktak, D. De Nie, and A. Jones. Methodological approach for evaluating higher tier pec groundwater scenarios supporting the prediction of environmental concentrations of pesticides at the pan european scale. In A. Del Re, M. Trevisan, and E. Capri, editors, *Pesticide chemistry symposium. Pesticide in air, plant soil and water system*, pages 951–962, 2003.

[18] I. Rivals and L. Personnaz. Mlps (mono-layer polynomials and multi-layer perceptrons) for non-linear modeling. *Journal of Machine Learning Research*, 3:1383–1398, 2003.

[19] J. Sacks, W. J. Welch, T.J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, 4(4):409–435, 1989.

[20] S. Shan and G. Wang. Development of adaptive rbf-hdmr model for approximating high dimensional problems. In *Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2009, San Diego, USA, DETC2009-86531*, 2009.

[21] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, 2003.

[22] A. Tiktak, JJTI Boesten, AMA Van der Linden, and M. Vanclooster. Mapping the vulnerability of european groundwater to leaching of pesticides with a process based meta-model of europearl. *Journal of Environmental Quality*, 35:1213–1226, 2006.

[23] A. Tiktak, D.S. De Nie, J.D. Pineros-Garcet, A. Jones, and M. Vanclooster. Assessing the pesticide leaching risk at the pan European level: the EuroPEARL approach. *Journal of Hydrology*, 289:222–238, 2004.

[24] D. J.J. Toal, N. W. Bressloff, and A. J. Keane. Kriging hyperparameter tuning strategies. *AIAA Journal*, 46(5):1240–1252, 2008.

[25] D.J.J. Toal, A. I.J. Forrester, N.W. Bresslof, A.J. Keane, and C. Holden. An adjoint for likelihood maximization. *Royal Society*, pre-print, 2009.

[26] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[27] B.A. Worley. Deterministic uncertainty analysis. *Nuclear Science and Engineering (ASCE)*, 94:180, 1986.

3 Surrogate-based Optimization

It is a good morning exercise for a research scientist to discard a pet hypothesis every day before breakfast. It keeps him young.
— Konrad Lorenz

3.1. Surrogate-Based Infill Optimization Applied to Electromagnetic Problems

I. Couckuyt, F. Declercq, T. Dhaene, H. Rogier

Published in Advances in design optimization of microwave/rf circuits and systems,

vol. 20, no. 5, pp. 492-501, September 2010.

Abstract

The increasing use of expensive computer simulations in engineering places a serious computational burden on associated optimization problems. Surrogate-based optimization becomes standard practice in analyzing such expensive black-box problems. This paper discusses several approaches that use surrogate models for optimization and highlights one sequential design approach in particular, namely, expected improvement. The expected improvement approach is demonstrated on two electromagnetic problems, namely, a microwave filter and a textile antenna.

3.1.1. Introduction

For many problems in science and engineering it is impractical to perform experiments on the physical world directly. Instead, complex, physics-based simulation codes are used to run experiments on computer hardware. While allowing scientists more flexibility to study phenomena under controlled conditions, computer experiments require a substantial investment of computation time. This is especially evident for routine tasks such as optimization, sensitivity analysis and design space exploration [42]. Regardless of the rapid advances in High Performance Computing (HPC) and multi-core architectures, it is rarely feasible to explore the complete design space using high-fidelity computer simulations. As a result researchers have turned to various approximation methods that mimic the behavior of the simulation model as closely as possible while being computationally cheap(er) to evaluate.

This work concentrates on the use of data-driven approximations using compact surrogate models (otherwise known as metamodels or response surface models). Examples of surrogate models include: rational functions, Gaussian Process (GP) models, and Support Vector Machines (SVM). It is crucial to stress the distinction between local and global surrogate modeling. With the latter, an approximation model of the output behavior of the simulator is built over the entire design space. By contrast, local surrogate models, often used in trust-region optimization frameworks, approximate only a small part of the design space and are discarded after use.

Most often, surrogate models are used to solve so-called forward problems. The practitioner is interested in the performance characteristics of a complex system, given the input parameters. The surrogate models create a mapping between the design space (input parameters) and the performance space (output parameters). In contrast, the focus of the reverse (inverse) problem is on exploring the design space. Hypothetically, a surrogate model could be created that maps the output parameters to the input parameters (as opposite to forward modeling) of the complex system over the entire design space. However, many inverse problems are typically ill-posed. Considering Hadamard's definition of ill-posedness [16], the two outstanding problems hampering the creation of a full inverse surrogate model are non-uniqueness and instability. A good overview of the associated intricacies is presented by Barton in [7]. For all the above reasons, the inverse problem is often reduced to the task of finding an input parameter combination for a certain output characteristic. Still, it is possible that,

1. no such input parameter combination exists
2. more than one input parameter combination satisfies the given output characteristic

A popular solution is to convert the inverse problem to a forward optimization problem, as is done in this paper.

The construction of highly efficient surrogate models is an entire research domain in itself. In order to arrive at an acceptable model, numerous problems and design choices must be overcome (what data collection strategy to use, what model type is most applicable, how should the model parameters be tuned, which variables are relevant, how to integrate problem-specific knowledge, etc.).

This paper describes a popular optimization method for expensive black-box simulators based on Kriging surrogate models, namely, expected improvement (EI) [21]. We provide a freely available implementation of the EI approach as a data collection (= sequential design) strategy in a flexible research platform for surrogate modeling, the **SU**rrogate **MO**deling (SUMO) Toolbox[1] [15]. The SUMO Toolbox is used to solve two complex problems both originating from Electromagnetics (EM). Previously, Kriging surrogate models have been used for EM device optimization by creating a global accurate Kriging surrogate model [30]. Afterwards, the computational cheap surrogate model is optimized instead of the expensive simulation. Although the EM device has been successful optimized, creating such one-shot Kriging surrogate models does not result in the most efficient use of expensive function evaluations. Siah et al. [40] try to minimize the number of function evaluations by applying the EI approach on two EM applications.

Section 3.1.2 provides related work of Surrogate-Based Optimization (SBO), including the expected improvement (EI) function. In Section 3.1.3, this EI approach is used to design and optimize an inter-digital filter (the forward problem). In Section 3.1.4, the material properties of a textile antenna are identified using EI (the inverse problem).

3.1.2. Surrogate-Based Optimization (SBO)

3.1.2.1. Introduction

SBO techniques are concerned with accelerating the optimization of expensive simulation problems. To speedup the optimization process other existing optimization algorithms have been adapted to minimize the number of function evaluations and to utilize parallel computing. A good overview is given in [2]. These existing optimization methods can still be significantly improved by taking advantage of surrogate models. The extra information provided by the surrogate models helps avoiding local optima and efficiently guides the search to the global optimum. Various directions have been undertaken to incorporate surrogate models in the optimization process.

In the context of evolutionary optimization surrogate models are used to provide a rough approximation to guide the global search, or a local accurate surrogate model is used to speedup the local search step, or a

[1] The SUMO Toolbox can be downloaded from: `http://sumo.intec.ugent.be`. An open source license will be available soon.

combination of both [35, 43]. For instance, Zhou et al. [44] apply a data parallel Gaussian process for the global approximation and a (simple) Radial Basis Function (RBF) model for the local search. Lim et al. [31] benchmark different local surrogate modeling techniques (quadratic polynomials, GP, RBF and extreme learning machine neural networks) including the use of (fixed) ensembles, in combination with evolutionary computation.

An important concept in global optimization is trust regions, introduced in surrogate modeling by [1]. Trust region-frameworks manage local surrogate models throughout the design space. A set of mathematics based [1] or pure heuristic [26] rules determines the size and location of the surrogate model. While trust region-frameworks are widely used in large scale optimization problems they have the disadvantage of sometimes overlooking the global optimum, as only a small part of the design space is approximated by the local surrogate model. On the other hand, by approximating only one local part of the design space at a time, it is possible to optimize very complex systems exhibiting non-linear behavior.

If a number of simulation models are available, each with varying accuracy (= fidelity), multi-fidelity methods [14], also known as variable-fidelity methods, can be used to solve more complex problems. There are several approaches to exploit multi-fidelity models. Without loss of generality, we can assume that only two simulation models are available, a low-fidelity and a high-fidelity model. An additive or multiplicative scaling factor [17] can be introduced based on a single (or a few) data point(s). The underlying idea is that these scaling factors correct the output of the low-fidelity model to agree with the output of the high-fidelity model near the vicinity of these points (zero-order scaling). One may also use higher order scaling strategies, e.g., where the derivatives of the low-fidelity model are also modified to agree with the high-fidelity model. A more complex combination of both approaches is also possible; for instance Eldred et al. [12] propose to write the low-fidelity model as a weighted combination of additive and multiplicative scaling factors. Alternatively, space mapping methods can be utilized. Instead of approximating the output space directly, space mapping [4, 3, 5, 29] maps the input space of a low-fidelity model to the input space of the high-fidelity model, basically employing an input correction between multiple fidelity models causing the optima to align in the design space. Recently it has been proposed to apply a similar technique for output correction, denoted by output space mapping [28, 6]. Moreover, combinations of input and output space mapping are also possible. In addition, recently [18] proposed a new variant called manifold mapping which can be seen as a generalization to output space mapping. Actually, the co-Kriging surrogate model [9, 25] is inherently a multi-fidelity surrogate model that essentially applies a correction to the output of the low-fidelity model. Multi-fidelity optimization methods, such as space mapping, are able to significantly improve on other methods by reducing computation time and/or generating better optimal designs. However, computational cheap low-fidelity models

may not always be available to the practitioner. In that case, one may turn to pure black-box methods to optimize expensive simulation codes.

3.1.2.2. Expected improvement

Another optimization approach is to use specific tailored adaptive sampling strategies while building global surrogate models. As the focus of the sampling algorithm is on optimization, global surrogate models are not necessarily accurate over the whole design space. In engineering, adaptive sampling strategies are also known as infill criteria. An infill criterion is a function, also known as figure of merit, that measures how *interesting* a data point is in the design space. Starting from an initial approximation of the design space, identifying new data points (infill or update points) to update the approximation model is then done by optimizing the infill criterion. In global SBO it is crucial to balance between exploration[2] and exploitation[3]. A well-known infill criterion that is able to effectively solve this trade-off is Expected Improvement (EI), which has been popularized by Jones et al. [21, 39] as the Efficient Global Optimization (EGO) algorithm. EI has been suggested in the literature as early as 1978 [34]. Jones wrote an excellent discussion regarding the infill criteria approach in [20]. Subsequently, Sasena compared different infill criteria for optimization and investigated extensions of those infill criteria for constrained optimization problems in [38].

The EI criterion can easily be interpreted graphically (see Figure 3.1). At $\mathbf{x} = 0.5$, a Gaussian probability density function (PDF) is drawn and expresses the uncertainty about the predicted function value of a sampled and unknown function $y = f(\mathbf{x})$. Thus, the uncertainty at any point $\mathbf{x}$ is treated as the realization of a random variable $Y(\mathbf{x})$ with mean $\hat{y} = \hat{f}(\mathbf{x})$ (= prediction) and variance $\hat{s}^2 = \hat{\sigma}^2(\mathbf{x})$ (= prediction variance). Assuming the random variable $Y(\mathbf{x})$ is normally distributed, then the shaded area under the Gaussian probability density function is the Probability of Improvement (PoI) of $Y(\mathbf{x})$ over the intermediate minimum function value f_{min}(the dotted line), denoted as $P(Y(\mathbf{x}) \leq f_{min})$, i.e.,

$$\begin{aligned} PoI(\mathbf{x}) = P(Y(\mathbf{x}) \leq f_{min}) &= \int_{-\infty}^{f_{min}} \phi(Y(\mathbf{x}))\, dY \\ &= \Phi\left(\frac{f_{min} - \hat{y}}{\hat{s}}\right), \end{aligned} \tag{3.1}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the normal probability density function and normal cumulative distribution function, respectively. The PoI is already a very

[2] enhancing the general accuracy of the surrogate model

[3] enhancing the accuracy of the surrogate model solely in the region of the (current) optimum

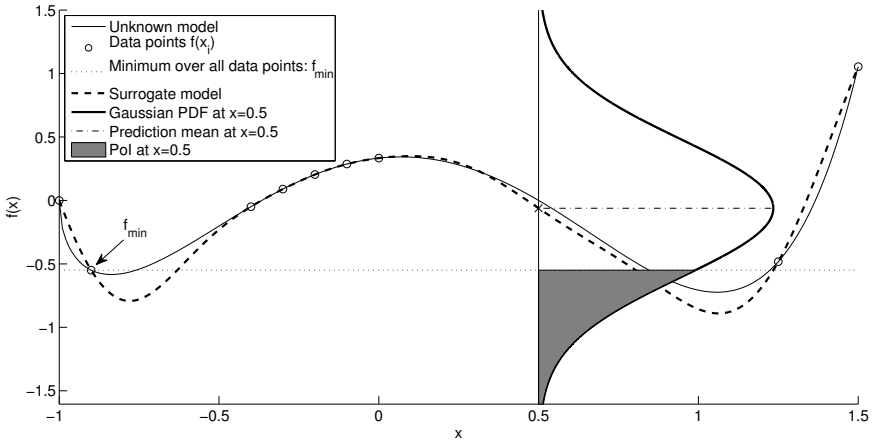


Figure 3.1.: Graphical illustration of a Gaussian process and expected improvement. A surrogate model (dashed line) is constructed based on some data points (circles). For each point the surrogate model predicts a Gaussian probability density function (PDF). An example of such a PDF is drawn at $x = 0.5$. The volume of the shaded area is the probability of improvement (PoI) and the first moment of this area is the expected improvement.

useful infill criterion. However, while this criterion describes the possibility of a better minimum function value, it does not quantify how large this improvement will be.

EI quantifies the improvement by considering the first moment of the shaded area, i.e., every possible improvement over f_{min} multiplied by the associated likelihood. For continuous functions EI is an integral defined as:

$$E[I(\mathbf{x})] = \int_{-\infty}^{f_{min}} I(\mathbf{x}) \cdot \phi(Y(\mathbf{x}))\, dY, \tag{3.2}$$

where

$$I(\mathbf{x}) = max(f_{min} - Y(\mathbf{x}), 0). \tag{3.3}$$

Hence, EI can be rewritten in closed form as:

$$E[I(\mathbf{x})] = \begin{cases} (f_{min} - \hat{y}) \cdot \Phi\left(\frac{f_{min} - \hat{y}}{\hat{s}}\right) + \hat{s} \cdot \phi\left(\frac{f_{min} - \hat{y}}{\hat{s}}\right) & if\ \hat{s} > 0 \\ 0 & if\ \hat{s} = 0 \end{cases}. \tag{3.4}$$

EI (Equation 3.4) and PoI (Equation 3.1) serve as utility functions, often conceived as figures of merit, which have to be optimized over $\mathbf{x}$ to find

the subsequent data point to evaluate. Note, however, that besides the prediction $\hat{y} = \hat{f}(\mathbf{x})$ of the surrogate model, a point-wise error estimation $\hat{s} = \hat{\sigma}(\mathbf{x})$ of the surrogate is also required.

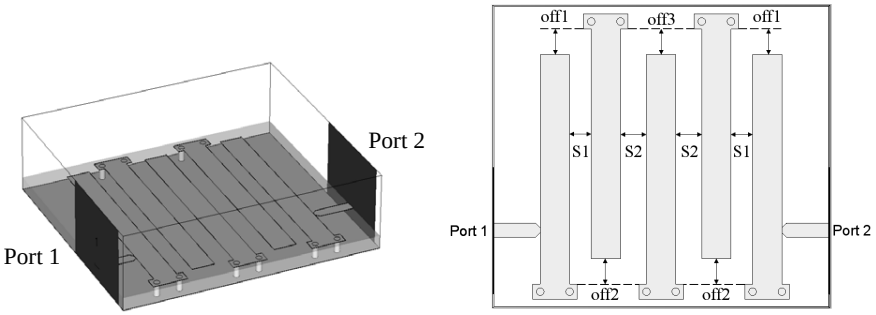
Therefore, the original EGO algorithm used Kriging [21] as surrogate model of choice, since Kriging provides analytical formulae for prediction as well as a point-wise error estimation. A full mathematical description of Kriging is beyond the scope for this paper. Kriging has been explained many times in the literature, hence, only a overview of the most influential papers is given here. A good starting point for Kriging are the introductions of Matheron et al. [33] and Sacks et al. [37]. Kriging, in fact, is part of a broader class of approximation methods, namely, Gaussian Processes (GP). While traditional approximation methods only predict a single function value, GP methods predict a complete normal distribution $Y(\mathbf{x}) \sim \mathcal{N}(\hat{y}, \hat{s})$ for each point $\mathbf{x}$. The predicted distribution imparts the probability that a particular function value occurs.

For a full overview of modern GP the reader is referred to the excellent GP reference book of Rasmussen et al. [36]. Depending on the context some authors coin the term Gaussian process (temporal) or Gaussian Random Field Metamodels (GRFM; spatial) [13], however, the underlying methods are the same.

3.1.3. Example 1: Microwave filter

3.1.3.1. Application

The first example is a microwave inter-digital filter, used for instance in cellular phones. This component can be analyzed in various ways. Circuit simulation allows for very fast evaluation with reasonable accuracy, whereas full-wave electromagnetic (EM) simulations provide high accuracy at a high computational cost. The inter-digital filter presented in this paper has been analyzed and optimized before by Swanson [41] using a combination of analytic methods, circuit simulation and EM simulations. We use the EI criterion to optimize the filter, and use the CST MicroWave Studio® (CST MWS) as a full-wave EM simulation tool. The top view of the filter is shown in Figure 3.2b and consists of five quarter-wavelength parallel microstrip resonators. The scalable layout is fully parametrized by $S1 = [32, 38]$ mm, $S1 = [40, 48]$ mm, $off1 = [-3, 9]$ mm, $off2 = [-3, 3]$ mm and $off3 = [-3, 3]$ mm. The last three parameters define the offset of the quarter-wavelength microstrip resonators with respect to the horizontal dashed lines. In particular, $off1$ is the offset of the outer two strips, $off2$ of the second and fourth strip and $off3$ of the middle strip. In other words, the offsets implicitly define the length of each microstrip. While $S1$ and $S2$ (spacings) denote the gap between the outer two strips and the inner two strips, respectively. This results in a symmetric structure for the filter. In total, this adds up to five geometric design variables that must be optimized.



(a) CST model.

(b) Top view. The five geometric design parameters define implicitly the length of the microstrips ($off1$, $off2$ and $off3$) and the spacings between the microstrips ($S1$ and $S2$).

Figure 3.2.: Microwave narrow-band filter.

The goal is to design a fifth-order ($N = 5$) narrow-band filter with a flat passband response centered around 2.44 GHz and with a 10% bandwidth. In case of a lossless structure, a specific relationship between the passband ripple and return loss of the filter allows us to minimize the ripple in the passband by minimizing the maximum of the S_{11}-parameter curve, i.e., the reflection coefficient, in the frequency range $[2.32, 2.56]$ GHz. No specific optimization goals were set for the insertion loss and the stopband. The 'fast S-parameter' solver in CST MWS is used, and a frequency sweep takes approximately 5 to 10 minutes on a standard laptop.

This optimization problem is used to benchmark several variants of Kriging surrogate modeling strategies in conjunction with the EI approach. In addition, this problem serves as an example that black-box SBO methods are able to find optimal designs that compare favorably with designs obtained using domain-specific knowledge [41].

3.1.3.2. Experimental setup

Version 6.1 of the SUMO toolbox is used to perform the optimization of the narrow-band filter and is configured as follows. The initial set of data points is generated by a maximin Latin Hypercube Design (LHD; implemented as in [23]) of 19 points together with 32 corner points, adding up to a total of 51 initial points. For this particular application the standard EI function (as defined in Section 3.1.2) is used to select infill points. The EI function is optimized using the DIviding RECTangles (DIRECT) algorithm of Jones et al. [22] to determine the next data point to evaluate. A time budget constraint of 24 hours is applied, i.e., the overall optimization process runs for 24 hours.

The aforementioned configuration is reproduced three times with different variants of the Kriging surrogate model. The first two cases configure Kriging as surrogate model of choice as implemented by the DACE toolbox [32]. More precisely, in one run the DACE toolbox itself performs the hyperparameter optimization, which comprises maximum likelihood estimation (MLE) using a modified Hooke & Jeeves direct search method [19] (pattern search). In another run, the hyperparameters of the Kriging model are identified by Matlab's Genetic Algorithm (GA) toolbox using 10-fold cross validation to guide the search. In the last configuration, a custom implementation of blind Kriging [24] is employed (MLE using the DIRECT algorithm).

In addition, the cost function is also optimized using the Matlab pattern search and simulated annealing routines using the default options and initial point $\mathbf{x_0} = (35, 44, 3, 0, 0)$. However, unlike the Kriging configurations, a time budget of 24 hours is not applied, instead the optimization is halted after exceeding the number of samples that the best Kriging configuration reached.

3.1.3.3. Results

Figure 3.3 shows the progress of the optimization process, i.e., the minimum cost function value versus the number of samples evaluated. The dotted line is the blind Kriging configuration and performs worst in terms of the final solution. Constructing a blind Kriging surrogate model is twice as expensive as standard Kriging, hence, less time is available to evaluate the expensive simulation code. On the other hand, Kriging (GA) is able to produce better Kriging models due to a large model parameter search with a genetic algorithm guided by cross validation, and, thus, that configuration finds attracting basins more quickly. However, due to the cost of cross validation and evolutionary-based strategies it is only able to evaluate approximately 190 samples before the time budget is exceeded. Yet, at that point it is the best performing method. The standard Kriging configuration finds the best solution. As it is significantly faster than the other two configurations, it is able to process more simulator runs, which proves to be more important in this application than a really accurate approximation model. After about 220 function evaluations it finds the best solution in the $5D$ design space and still has time to select about 100 more samples to look for an even better solution or to validate the current one. It should be noted that the usefulness of more expensive surrogate modeling strategies (such as blind Kriging) may improve when the time of a single simulation run would increase to hours or even days. The final solutions found of each technique are displayed in Table 3.1, together with the reference optimum, as found by Swanson [41].

Surprisingly, the design found using the EI criterion and the Kriging (MLE) surrogate models outperforms the reference design, i.e., with respect

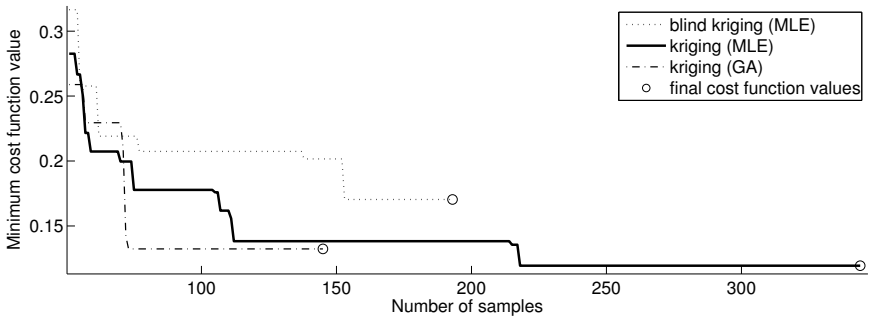


Figure 3.3.: Evolution of the minimum cost function value versus the number of samples evaluated in 24 hours. The standard Kriging (MLE) surrogate model finds the lowest cost function value. (inter-digital filter)

method	$\lvert X\rvert$	$\mathbf{x_{min}}$	f_{min}
blind Kriging (MLE)	193	$(35.03, 41.37, 8.99, -1.14, 0.19)$	0.17038
Kriging (GA)	145	$(36.00, 43.10, 5.90, -3.00, -3.00)$	0.13234
Kriging (MLE)	344	$(36.68, 44.16, 6.16, -2.67, -2.47)$	0.11936
pattern search	344	$(36.00, 43.00, 7.56, -1.77, -1.44)$	0.13426
simulated annealing	344	$(37.12, 43.10, 8.39, -1.87, 0.42)$	0.18074
reference [41]	*unknown*	$(37.10, 44.46, 6.30, -2.60, -2.43)$	0.12527

Table 3.1.: Final designs of the inter-digital filter. $\lvert X\rvert$ is the number of samples evaluated (in 24 hours), $\mathbf{x_{min}}$ and f_{min} are the final solutions and cost function values respectively (inter-digital filter).

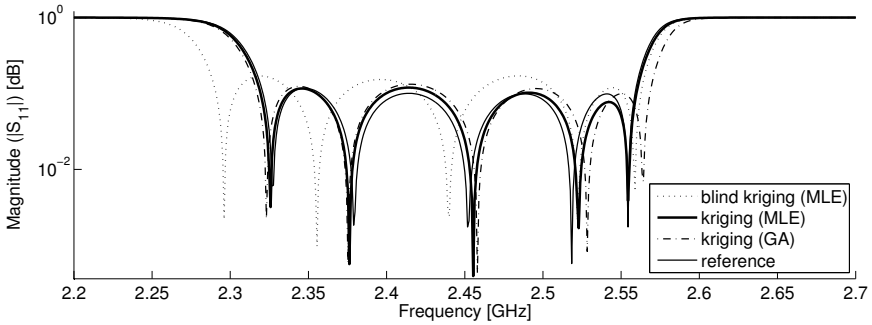


Figure 3.4.: S_{11}-parameter magnitude plots of the different solutions. (inter-digital filter)

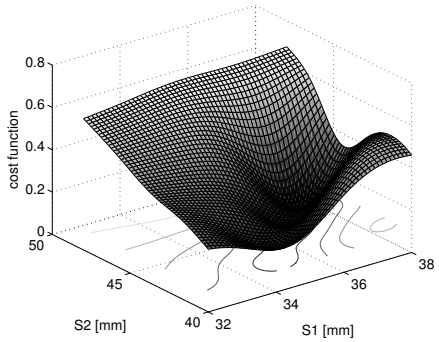


Figure 3.5.: 2D slice plot of the Kriging (MLE) surrogate model of the cost function. The offset parameters are set to the values of the final solution, i.e., $off1 = 6.16$ mm, $off2 = -2.67$ mm and $off3 = -2.47$ mm. (inter-digital filter)

to the same cost function. To compare the different designs in a fair way the S_{11}-parameter curves are drawn in Figure 3.4. The reference design is constructed so that the ripples in the S_{11}-parameter curve are of equal height (exact equal ripple tuning). This might also be important as it guarantees consistent performance over the whole frequency range of interest. Note however that in this paper the cost function does not punish or favor equal ripple tuning. Thus, the solution found in this paper is better on the cost function but has not exact equal ripples (though very close to), while the reference design is slightly more consistent over the frequency range with regard to the ripples.

A huge advantage of SBO is the ability to easily explore the final (and intermediate) approximation models. The practitioner is able to cheaply analyze the robustness of the solution, locate other interesting regions (e.g.,

local optima), etc. For illustration purposes the Kriging (MLE) surrogate model of the cost function is shown in Figure 3.5. This plot is a $2D$ slice of the $5D$ design space where the offset parameters are fixed, i.e., $off1 = 6.16$ mm, $off2 = -2.67$ mm and $off3 = -2.47$ mm.

3.1.4. Example 2: Textile antenna

3.1.4.1. Application

Material property identification is a well-known inverse problem. In particular, we address the characterization of the electrical properties of textile materials for the use in textile antennas. These antennas are constructed out of a non conductive textile substrate, a conductive ground plane and a conductive patch as shown in Figure 3.6. The textile substrate used here has a vegetable cellulose based origin and has a thickness of 0.805 mm. The textile substrate's electrical properties of interest are the permittivity ϵ_r and loss tangent $\tan\delta$, and together with the patch geometry they determine the antenna performance indicators such as resonance frequency and bandwidth. The goal is to accurately characterize these two material properties of the textile antenna substrate based on the measured performance characteristics, provided by the reflection coefficient measurement of the antenna. The exact knowledge of the textile substrate's properties will then be exploited in the computer-aided design of complex wearable antenna topologies. Previously, manual fitting of the simulated and measured data has been reported in [11] for extracting the permittivity and loss tangent of the textile substrate. Based on a rough estimation (rule of thumb) of the textile substrates' electrical properties a full-wave EM simulation is performed to design a single-mode textile antenna with a sharp resonance. Therefore, the length L of the patch antenna is chosen such that a resonance is obtained in the vicinity of 2.4 GHz. Impedance matching is obtained by optimizing the width W and the coaxial feed positions x_f and y_f. The resulting patch dimensions L, W and feed positions x_f and y_f, based on the estimated permittivity and loss tangent of the textile substrate are 45.5 mm, 33 mm, 11 mm and 16.5 mm respectively. The way the real resonance peak of the textile antenna has shifted and changed in form allows determining the actual permittivity and loss tangent of the substrate. Therefore, the textile antenna's reflection coefficient is measured and compared to simulations for multiple substrate parameters using ADS Momentum in the $[2, 3]$ GHz frequency range.

As a full inverse surrogate model is infeasible, the inverse problem is converted to a forward optimization problem. Specifically, the problem is reduced to the minimization of an error function (= cost function) between the simulation results $\mathbf{y}$ and the measured data $\tilde{\mathbf{y}}$ (see Figure 3.7). The error function is the popular Mean Squared Error (MSE) defined by,

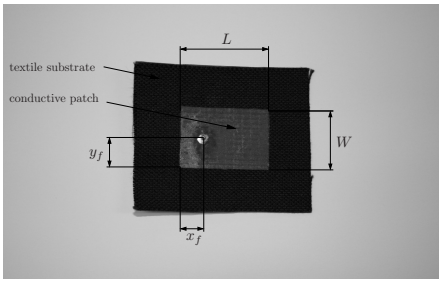


Figure 3.6.: Photograph of the textile antenna.

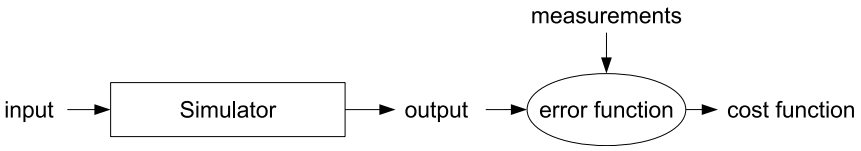


Figure 3.7.: The inverse problem is solved by minimizing the error function between the simulation results $\mathbf{y}$ and the measured data $\tilde{\mathbf{y}}$.

$$MSE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2, \tag{3.5}$$

with $n = 401$ the number of frequency points (samples). In the optimization process ϵ_r and $\tan\delta$ are bounded by $[1.1, 2.5]$ and $[0.020, 0.090]$, respectively. Hence, the correct material properties are identified by minimizing the MSE between the magnitude (in dB) of the simulated S_{11} curve and the measured S_{11} curve.

A difficult problem often encountered with inverse problems is the presence of noise in the cost function. Obviously, the reflection coefficient measurements used in the error function contain noise. The error function (MSE) reduces the noise in the cost function. Any remaining noise is handled by the Kriging surrogate model.

3.1.4.2. Experimental setup

Version 6.2 of the SUMO toolbox is utilized to solve the inverse problem of the textile antenna. The configuration is quite similar to the previous (forward) problem. An initial set of samples is generated by an optimal maximin Latin Hypercube Design (LHD; [10]) of 10 points together with four corner points, adding up to a total of 14 initial points. Subsequently, infill points are selected using the EI as a figure of merit which is optimized by the DIRECT algorithm. The optimization is halted when the number of samples exceeds 70.

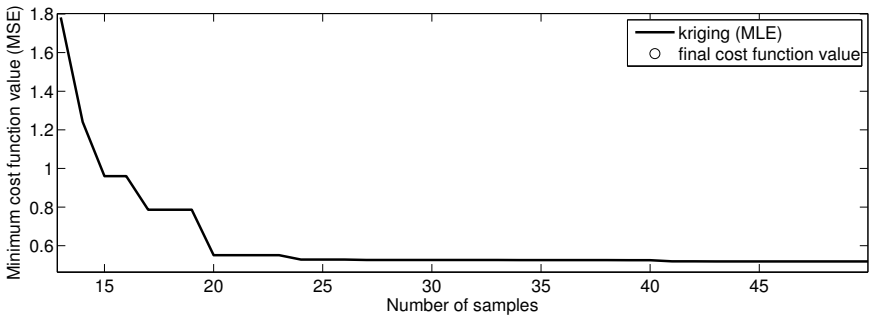


Figure 3.8.: Evolution of the minimum cost function value versus the number of samples evaluated. The EI approach quickly locates the region of the global optimum and, subsequently, explores this region further, fine-tuning the final solution. (textile antenna)

The surrogate model of choice is a custom implementation of Kriging. The hyperparameters are determined using Maximum Likelihood Estimation (MLE). The actual optimization is accomplished by a Sequential Quadratic Programming method (SQPLab[4] [8]), taking into account derivative information. Finally, Kriging is modified to approximate the (noisy) cost function, instead of using interpolation.

To provide a comparison against traditional black-box optimization techniques the cost function is also optimized using the Matlab pattern search and simulated annealing routines using initial point $\mathbf{x_0} = (1.80, 0.05)$ and the same sample budget as the Kriging configuration. The remaining options are left to their default values.

3.1.4.3. Results

An evolution plot of the minimum cost function values versus the number of samples is depicted in Figure 3.8. Starting from 14 samples the EI criterion quickly locates the region of the global optimum. At 20 samples the EI function starts exploring other parts of the design space (the flat parts), occasionally fine-tuning the current solution At approximately 45 samples the design space has been sufficiently explored and the final solution has been found. Still, the sampling continues until the sample budget is met, though no improvement is made.

The final optimal parameter combinations of each technique are presented in Table 3.2 along with the solution obtained through manual fitting and experimentation. The S_{11} curves of the optimal simulation run and the measurements are plotted in Figure 3.9. The solution found in this paper

[4]SQPLab is found at `http://www-rocq.inria.fr/~gilbert/modulopt/optimization-routines/sqplab/sqplab.html`

method	$\|X\|$	ϵ_r	$\tan\delta$	MSE
Kriging (MLE)	71	1.691	0.054	0.5185
pattern search	71	1.675	0.056	0.8963
simulated annealing	71	1.705	0.065	1.1513
reference	*unknown*	1.694	0.060	0.6974

Table 3.2.: Final material parameters. $|X|$ is the number of samples, ϵ_r and $\tan\delta$ are the material parameters with the associated cost function value (MSE) (textile antenna).

is significantly better than the reference optimum with respect to the cost function.

Finally, the final Kriging surrogate model of the cost function is displayed in Figure 3.9. As can be seen in the contour plot, the EI function explored the edges of the design space quite thoroughly, increasing the accuracy of the Kriging model. Afterwards, more attention is paid to the valley, sampling densely near the region of the global optimum (the cluster of points).

3.1.5. Conclusion and future work

This work provides an overview of several Surrogate-Based Optimization (SBO) methods. A SBO approach based on the Expected Improvement (EI) criterion is implemented in a freely available toolbox, namely, the SUMO toolbox. The SUMO toolbox is used to benchmark different variants of the Kriging surrogate model on a forward EM optimization problem. Subsequently, a novel inverse EM problem is solved by minimizing the error between simulated data and measured data.

The optimization results of the first application show standard Kriging (MLE) to outperform other variants of Kriging. In fact, it is demonstrated that the obtained design compares well against a reference design obtained by a domain expert.

The material property identification (inverse) problem is solved by optimizing an error function. The Kriging surrogate model, adapted for regression, is able to approximate the noisy cost function very accurately, resulting in the identification of the material parameters with a minimal number of expensive function evaluations. The final material parameter combination largely agrees with the measured data. In particular, the identification of the resonance peak of the S_{11} curve is highly accurate. Any remaining difference between the measured data and the best solution is due to missing parameters in the simulation, e.g., the finite conductivity of the conductive parts is not taken into account.

Future work includes full inverse electrical characterization of textile ma-

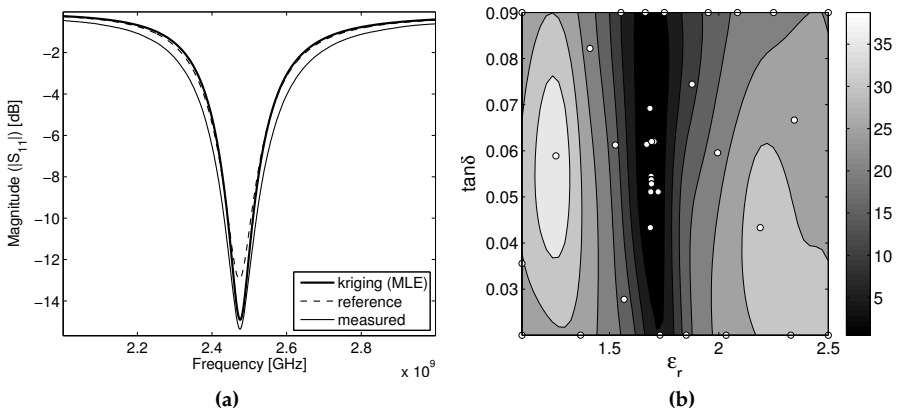


Figure 3.9.: (a) S_{11}-parameter magnitude plots of the identified material parameter combinations. The design found by Kriging (MLE) is significantly more accurate than the reference optimum, being able capture the measured resonance peak nicely. (b) Contour plot of the final Kriging (MLE) surrogate model of the cost function based on 71 data points. (textile antenna)

terials by including the finite conductivity and extensions to MultiObjective Surrogate-Based Optimization (MOSBO) [27] methods.

3.1.6. Bibliography

[1] N. Alexandrov, J. E. Dennis, R. M. Lewis, and V. Torczon. A trust region framework for managing the use of approximation models in optimization. *Structural Optimization*, 15:16–23, October 1998.

[2] S. Andradóttir. A review of simulation optimization techniques. In *Proceedings of the Winter Simulation Conference*, pages 151–158. IEEE Computer Society Press, 1998.

[3] M. Bakr, J.W. Bandler, K. Madsen, and J. Sondergaard. Review of the space-mapping approach to engineering optimization and modeling. *Optimization and Engineering*, 1(3):241–276, 2000.

[4] J.W. Bandler, R. M. Biernacki, Shao Hua Chen, P. A. Grobelny, and R. H. Hemmers. Space mapping technique for electromagnetic optimization. *IEEE Transactions on Microwave Theory and Techniques*, 42(12):2536–2544, Aug. 1994.

[5] J.W. Bandler, Q.S. Cheng, S.A. Dakroury, A.S. Mohamed, M.H. Bakr, K. Madsen, and J. Sóndergaard. Space mapping: the state of the art.

IEEE Transactions on Microwave Theory and Techniques, 52(1):337–361, Jan. 2004.

[6] J.W. Bandler, D. M. Hailu, K. Madsen, and F. A. Pedersen. Space-mapping interpolating surrogate algorithm for highly optimized EM-based design of microwave devices. *IEEE Transactions on Microwave Theory and Techniques*, 52:2593–2600, 2004.

[7] R.R. Barton. Issues in development of simultaneous forward-inverse metamodels. In *Proceedings of the Winter Simulation Conference*, pages 209–217, 2005.

[8] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2006.

[9] Noel A. C. Cressie. *Statistics for spatial data*. John Wiley & Sons, 1993.

[10] E.R. Dam, B.G.M. van Husslage, D. den Hertog, and J.B.M. Melissen. Maximin Latin hypercube designs in two dimensions. *Operations Research*, 55(1):158–169, 2007.

[11] F. Declercq, H. Rogier, and C. Hertleer. Permittivity and loss tangent characterization for garment antennas based on a new matrix-pencil two-line method. *IEEE Transactions on Antennas and Propagation*, 56(8):2548–2554, 2008.

[12] M.S. Eldred, A.A. Giunta, and S.S. Collis. Second-order corrections for surrogate-based optimization with model hierarchies. In *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis & Optimization Conference*, 2004.

[13] M.T.M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[14] A. I.J. Forrester and A. J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45:50–79, 2009.

[15] D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene. Sequential modeling of a low noise amplifier with neural networks and active learning. *Neural Computing and Applications*, 18(5):485–494, Jun. 2009.

[16] J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. Technical Report 49–52, Princeton University Bulletin, 1902.

[17] R.T. Haftka. Combining global and local approximations. *AIAA Journal*, 29:1523–1525, 1991.

[18] P.W. Hemker and D. Echeverría. A trust-region strategy for manifold-mapping optimization. *Computational Physics*, 224(1):464–475, 2007.

[19] R. Hooke and T. A. Jeeves. "direct search" solution of numerical and statistical problems. *J. of the Association of Computing Machinery (ACM)*, 8(2):212–229, 1961.

[20] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Global Optimization*, 21:345–383, 2001.

[21] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[22] D.R. Jones, C.D. Perttunen, and B.E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Optimization Theory and Applications*, 79(1):157–181, 1993.

[23] V. R. Joseph and Y. Hung. Orthogonal-Maximin Latin hypercube designs. *Statistica Sinica*, 18:171–186, 2008.

[24] V. R. Joseph, Y. Hung, and A. Sudjianto. Blind kriging: A new method for developing metamodels. *ASME Journal of Mechanical Design*, 130(3):031102–1–8, 2008.

[25] M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87:1–13, 2000.

[26] F. V. Keulen and V. V. Toropov. The multipoint approximation method in a parallel computing environment. *Journal of Applied Mathematics and Mechanics*, 79:67–70, 1999.

[27] J. Knowles and H. Nakayama. Meta-modeling in multiobjective optimization. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 245–284. Springer-Verlag, Berlin, Heidelberg, 2008.

[28] S. Koziel and J.W. Bandler. Interpolated coarse models for microwave design optimization with space mapping. *IEEE Transactions on Microwave Theory and Techiques*, 55:1739–1746, 2007.

[29] S. Koziel and J.W. Bandler. Space-mapping optimization with adaptive surrogate model. *IEEE Transactions on Microwave Theory and Techniques*, 55(3):541–547, March 2007.

[30] L. Lebensztajn, C.A.R. Marretto, M.C. Costa, and J-L Coulomb. Kriging: a useful tool for electromagnetic device optimization. *IEEE Transactions on Magnetics*, 40(2):1196–1199, 2004.

[31] D. Lim, Y-S. Ong, Y. Jin, and B. Sendhoff. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 07)*, pages 1288–1295, New York, NY, USA, 2007. ACM.

[32] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.

[33] G. Matheron. Principles of geostatistics. *Economic Geology*, 58:1246–1266, 1963.

[34] J. Mockus, V. Tiesis, and A. Zilinskas. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

[35] Y-S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *American Institute of Aeronautics and Astronautics Journal*, 41(4):687–696, 2003.

[36] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[37] J. Sacks, W. J. Welch, T.J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, 4(4):409–435, 1989.

[38] M.J. Sasena. *Flexibility and Efficiency Enhancements For Constrainted Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

[39] M. Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, 1997.

[40] E.S. Siah, T. Ozdemir, J.L. Volakis, P. Papalambros, and R. Wiese. Fast parameter optimization of large-scale electromagnetic objects using DIRECT with Kriging metamodeling. *IEEE Transactions on Microwave Theory and Techniques*, 52(1):276–285, 2004.

[41] D. G. Swanson. Narrow-band microwave filter design. *IEEE Microwave magazine*, 8:105–114, 2007.

[42] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[43] Z. Z. Zhou, Y. S. Ong, M. H. Lim, and B. S. Lee. Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Soft Computing*, 11(4):957–971, 2007.

[44] Z. Z. Zhou, Y-S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man and Cybernetics - Part C*, 37(1):66–76, 2007.

3.2. Automatic Surrogate Model Type Selection During the Optimization of Expensive Black-box Problems

I. Couckuyt, D. Gorissen, F. De Turck, T. Dhaene

Published in Proceedings of the Winter Simulation Conference,

pp. 4269-4279, 2011

Abstract

The use of Surrogate-Based Optimization (SBO) has become commonplace for optimizing expensive black-box simulation codes. A popular SBO method is the Efficient Global Optimization (EGO) approach. However, the performance of SBO methods critically depends on the quality of the guiding surrogate. In EGO the surrogate type is usually fixed to Kriging eventhough this may not be optimal for all problems. In this paper the authors propose to extend the well-known EGO method with an automatic surrogate model type selection framework that is able to dynamically select the best model type (including hybrid ensembles) depending on the data available so far. Hence, the expected improvement criterion will always be based on the best approximation available at each step of the optimization process. The approach is demonstrated on a structural optimization problem, i.e., reducing the stress on a truss-like structure. Results show that the proposed algorithm consequently finds better optimums than traditional Kriging-based infill optimization.

3.2.1. Introduction

Surrogate-Based Optimization (SBO) is an important research domain concerned with accelerating the optimization of expensive simulation problems [4, 15]. In SBO intermediate surrogate models, also known as metamodels, are built to estimate the objective function in consecutive iterations.

A popular SBO approach is to use global surrogate models and emphasize on adaptive sampling, or infill criteria [10]. Starting from an initial low-fidelity surrogate model based on a limited set of samples spread over the complete design space, the infill criterion identifies new samples of interest (infill or update points) to update the surrogate model. It is crucial in global SBO to strike a correct balance between exploration - enhancing the general accuracy of the surrogate model - and exploitation - enhancing the accuracy of the surrogate model solely in the region of the (current)

optimum. A well-known infill criterion that is able to effectively solve this trade-off is Expected Improvement (EI). The EI criterion has been suggested in literature as early as 1978 [19], and has been popularized by Jones et al. [11] in the Efficient Global Optimization (EGO) algorithm. An interesting discussion of the infill criteria approach is given by Jones [10].

While the EI approach is proven to be an efficient figure of merit, the quality of the surrogate model is still arguably the most important factor in the optimization process. In the EGO algorithm the surrogate model type of choice is the Kriging model as it provides the prediction variance required by EI. However other surrogate model types such as Support Vector Machines (SVM), pure Gaussian Processes (GP), Radial Basis Functions (RBF), etc. are also possible and may have superior accuracy for some problems. Unfortunately it is rarely possible to choose the optimal surrogate model type upfront as the behavior of the objective function is often poorly understood or even unknown.

In this paper the authors propose to combine an Evolutionary Model Selection (EMS) algorithm with the well-known EI criterion, see Section 3.1.2.2 on page 3-5. The EMS algorithm dynamically selects the best performing surrogate model type at each iteration of the EI algorithm. Thus, each iteration a new expensive sample point is chosen based on the EI criterion, which in its turn is based on the best surrogate model found by the EMS algorithm. This methodology is compared against traditional Kriging-based infill optimization on a structural dynamics problem, namely, the optimization of a truss structure. Note that the simulation code of the truss structure is deterministic, in contrast to stochastic simulation.

Section 3.2.2 summarizes related work on automatic model selection and SBO. Subsequently, in Section 3.2.3, the EMS methodology is described. Details of the application are found in Section 3.2.4, while the experimental setup is described in Section 3.2.5. Results and conclusion form the last two sections of this paper, i.e., 3.2.6 and 3.2.7.

3.2.2. Related work

Automatic model selection approaches for a single model type are quite common: [31, 3, 5, 16, 30, 27]. Integration with adaptive sampling has also been discussed [2]. However, these efforts do not tackle the surrogate model *type* selection problem, as they restrict themselves to a particular model type (e.g., SVMs or neural networks). As [15] states, "*Little is known about which types of model accord best with particular features of a landscape and, in any case, very little may be known to guide this choice.*". Likewise, [25] notes: "*...it is important to stress that there are always situations when one model type cannot be applied or suffers from inadequacies and can be well complemented or replaced by another one*". Thus an algorithm to solve this problem in a dynamic, fully automated way is very useful [14].

Voutchkov et al. [29] compare different surrogate model types for approximating multiple objectives during optimization. Similar work has been done by [21]. A lot of work concerning the simultaneous use of multiple surrogate model types has been done in the context of evolutionary optimization [20, 32]. Lim et al. [18] benchmark different local surrogate modeling types (quadratic polynomials, GP, RBF and extreme learning machines neural networks) including the use of (fixed) ensembles, for optimization of computationally expensive simulation codes. However, these approaches still require an a priori choice of model type and does not allow any dynamic switching between the model types. Other work consists of constructing a separate surrogate model for the global search and the local search. For instance, Zhou et al. [33] apply a Data Parallel Gaussian process for the global approximation and a simple RBF model for the local search.

A different approach is taken by Lim et al. [17] who approximate the objective function by a weighted ensemble where the weights are dynamically chosen according to the accuracy of each surrogate model, in effect adapting the surrogate model type for the problem at hand. In parallel, an independent search is applied using a simple polynomial model to account for smoothing. The work by Sanchez et al. [22] and Goel et al. [6] is similar. Both provide new algorithms for generating an optimal set of ensemble members for a fixed set of data points (no sampling).

The EI approach has primarily been applied separately in conjunction with Kriging models [11] and RBF models [23, 24]. Recently Viana et al. included other type of surrogate models [28]. In this paper we demonstrate how the EMS algorithm described in [8] can be seamlessly integrated into the EI approach to allow dynamic switching of the surrogate model type and to allow the use of ensembles, see Figure 3.10a.

3.2.3. Evolutionary model selection

The Evolutionary Model Selection (EMS) algorithm is based on a Genetic Algorithm (GA) with speciation (using the island model). We restrict ourselves to a brief overview of the EMS algorithm, a detailed treatment can be found in [8].

An initial sub-population (deme) is created for each surrogate model type and each deme is allowed to evolve according to an elitist GA. The different surrogate model types are implemented as Matlab objects (with full polymorphism) and each surrogate model type can choose its own representation and genetic operator implementations. This is important since it allows the genetic operators to be fully customized for each surrogate model type, allowing domain knowledge to be exploited, and improving the search efficiency. The GA is driven by a fitness function that calculates the quality of the surrogate model fit on the data. Parents are selected according to a selection algorithm (e.g., tournament selection) and offspring are generated through mutation and recombination genetic operators. The

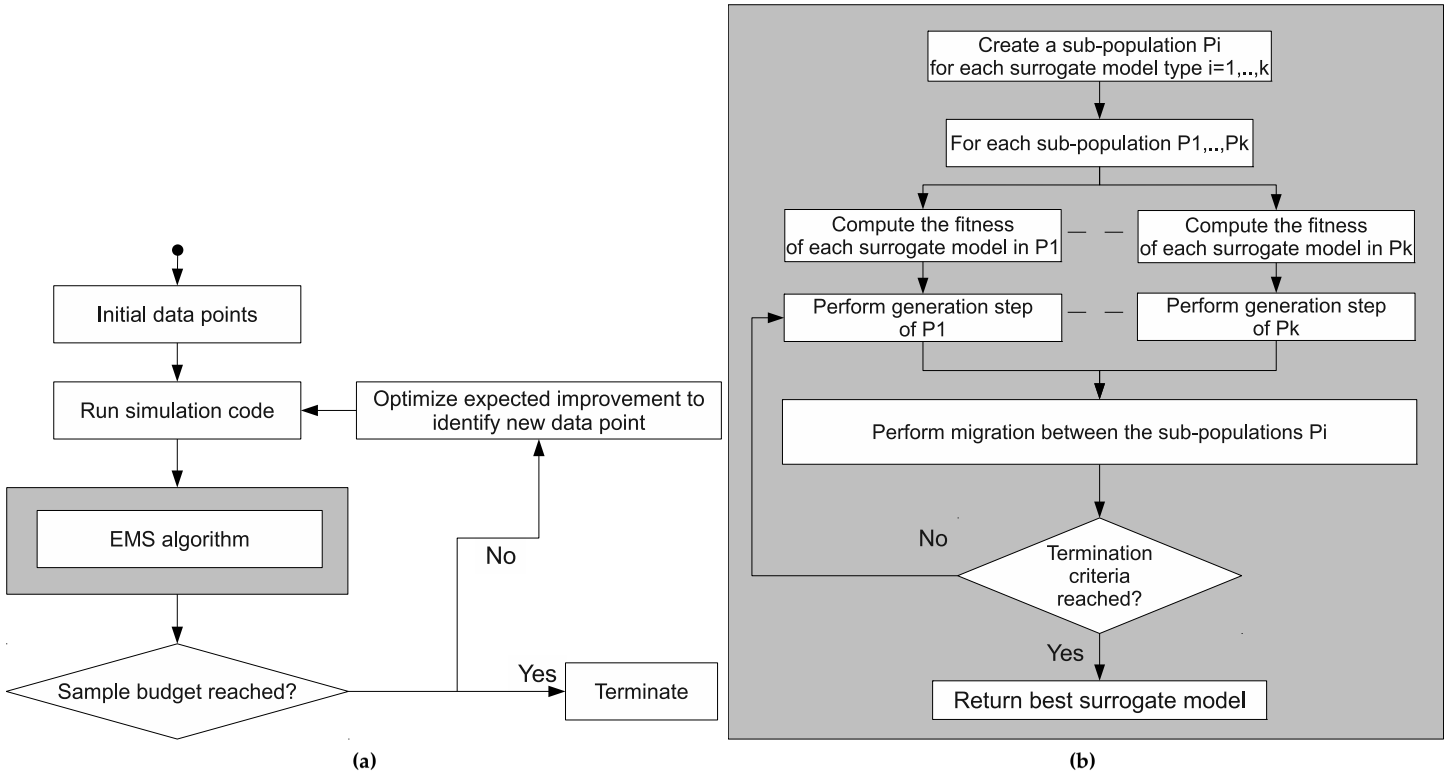


Figure 3.10.: a) Flow chart of the expected improvement method coupled with the EMS algorithm. After creating and evaluating the initial data points the EMS algorithm identifies the surrogate model that has the best accuracy (for the given dataset). Subsequently, a new data point is selected by optimizing the expected improvement. b) Flow chart of the Evolutionary Model Selection (EMS) algorithm. For each surrogate model type under consideration a separate sub-population is created. Each sub-population creates offspring using (surrogate model type specific) genetic operators (the generation step). It is during the generation step that different surrogate model types may be combined into ensemble models. Afterwards migration occurs between the sub-populations.

current deme population is then replaced with its offspring together with k elite individuals. Once every deme has gone through a generation, migration between individuals is allowed to occur at migration interval m_i, with migration fraction m_f and migration direction m_d (a ring topology is used). The migration strategy is as follows: if p is the population size of each deme, then the $l = (p * m_f)$ fittest individuals of the *i-th* deme replace the l worst individuals in the next deme (defined by m_d). Migrants are duplicated, not removed from the source population. Thus migration ensures competition between model types since it causes model types to mix. Models which a higher fitness (e.g., better accuracy) will tend to have a higher chance of propagating to the next generation. See Figure 3.10b for a general overview of the EMS algorithm.

An important consequence of migration is that the recombination operator (crossover) may occur on two surrogate models of different type (e.g., a rational function and a Support Vector Machine). Since a meaningful crossover between two different model types is not possible on the genotype level, the EMS implementation uses a behavioral crossover operator. The two different model types are merged together into an ensemble. Ensemble models that arise in the population through such heterogeneous crossovers, are simply treated as an additional model type (with its own operators and representation) and propagate through the population just as the other model types.

The EMS algorithm will iterate until some stopping criteria has been reached (e.g., model accuracy below 1%, or maximum number of generations exceeded). In addition an extinction prevention algorithm is used to ensure no surrogate model type is driven completely extinct.

3.2.4. Problem Application

The proposed method is applied to a structural dynamics problem. The problem is the optimal design of a two-dimensional truss, constructed by 42 Euler-Bernoulli beams. The goal is to identify a design that is optimal (or close to) with respect to passive vibration isolation. The truss structure is shown in figure 3.11 and is a simplification of a truss type typically used in satellites. Furthermore, the truss simulation code is deterministic, i.e., repeated simulations return exactly the same performance (in contrast to stochastic simulation).

The beams consist each of two finite elements and are subject to a unit force excitation at node one across a $100 - 200$Hz frequency range. The two leftmost nodes are fixed (cantilevered nodes) and all the other nodes are free to move around. There are four input parameters defining the position of nodes nine and ten in the structure and one output parameter, namely, the stress that the outermost node (the tip) receives. Thus, the geometry of the structure is varied by allowing nodes nine and ten to move inside 0.9×0.9 squares while the remaining nodes are set fixed (see Figure 3.11).

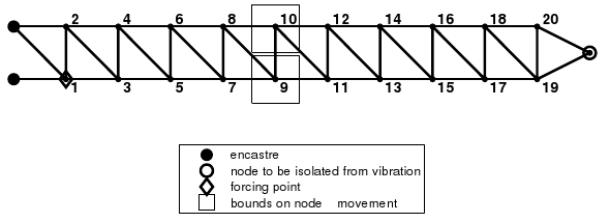


Figure 3.11.: Truss structure consisting of 42 beams. The stress is measured at the outermost (right) node, while the two left-most nodes are fixed.

The objective is to maximize the band-averaged vibration attenuation at the tip compared to the baseline structure (= objective function). To give an idea of the complexity of the optimization problem, a 2D slice plot of the objective function is seen in Figure 3.13a. For an in-depth discussion of the problem the reader is referred to Keane et al. [13].

3.2.5. Experimental setup

Version 6.2.1 of the SUMO Toolbox [7] is used to optimize the truss structure and is configured as follows. The initial set of samples in the $4D$ design space is generated by an optimal maximin Latin Hypercube Design (LHD; [9]) of 20 points augmented with 16 corner points, adding up to a total of 36 initial points. Several variants of the EI criteria are available in the toolbox, but for this particular application the original EI function is used to select infill points. The EI function is optimized using the DIviding RECTangles (DIRECT) algorithm of Jones et al. [12] to determine the next sample point to evaluate. Whenever the DIRECT algorithm is unable to obtain an unique sample a fallback criterion is optimized. This fallback criterion represents the Mean Squared Error (MSE) between the current best surrogate model and all the previous surrogate models that have been stored in the history. In effect, this identifies the locations in the domain where the prediction of surrogate models disagree the most.

The aforementioned configuration is repeated three times with different surrogate modeling strategies. The first two cases use Kriging as the surrogate model type of choice. More precisely, in the first run the hyperparameters are obtained through maximum likelihood estimation (MLE) using SQPLab [1] (utilizing likelihood derivative information). In the second run, the hyperparameters of the Kriging model are identified by Matlab's GA toolbox guided by 5-fold cross validation. In the third run the surrogate model is produced by the EMS algorithm as described in Section 3.2.3. Remark that since the EI approach is used for optimization, only model types that support a point-wise error estimation (= prediction variance) can be

included in the EMS method. Thus, the surrogate model types that compete in the evolution are Kriging models, RBF and Least Squares-Support Vector Machines (LS-SVMs; using [26]). This means that, in addition to the ensemble models (that arise as a result of a crossover between two different model types), four model types will compete to fit the objective function. For this paper a straightforward weighted ensemble model is used where all weights are set equal. This was chosen since it is simple to implement and understand, and introduces no additional parameters. More complex ensemble methods such as bagging or the method used by Goel et al. [6], can easily be incorporated. The population size for each model type is 10 and the maximum number of generations between each sampling iteration is 15. The maximum ensemble size is set to four. The final population of the previous model type selection run is used as the initial population for the next iteration. The optimization of the EI is applied on the best performing surrogate model type as determined by the EMS algorithm and 5-fold cross validation. The error function that is minimized is the Root Relative Squared Error ($RRSE$),

$$RRSE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n}\sqrt{\frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{\mathbf{y}})^2}}.$$

Each of the three different cases is allowed to run for 350 samples, in other words, the optimization process halts after 350 calls to the simulator. Finally, the tests are repeated 20 times to smooth out random effects.

3.2.6. Results

The average minimum objective function value at each iteration of the EI approach is shown in Figure 3.12. It can be seen that the EMS algorithm is consistently better than the other two surrogate modeling strategies, though the difference is quite small. Worth mentioning is that the evolution of the minimum function value of the EMS algorithm is smoother than, for instance, Kriging (MLE). For the latter, the function value clearly decreases in a stepwise fashion, while for the EMS algorithm other surrogate model types than Kriging step in whenever they are better in approximating the intermediate set of samples, thus, a better function value is found more often.

A Box-and-whiskers plot of the final function value for the three surrogate modeling strategies is shown in Figure 3.13b. Kriging (GA) and the EMS algorithm are substantially more stable than using Kriging (MLE), where the θ parameters are optimized using the likelihood. The EMS algorithm is also an improvement over Kriging (GA) as the quantiles lie closer to the smallest function value found.

Further analysis of the EMS algorithm is rather interesting. The final best surrogate model is almost always an ensemble model, with only one case

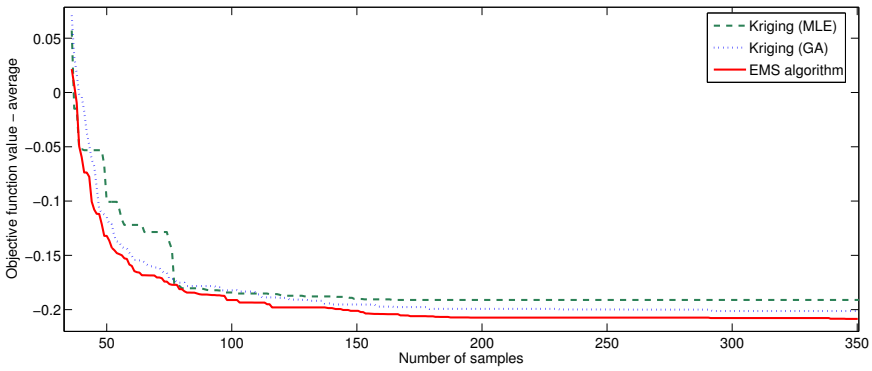


Figure 3.12.: Evolution of the number of samples versus the minimum value.

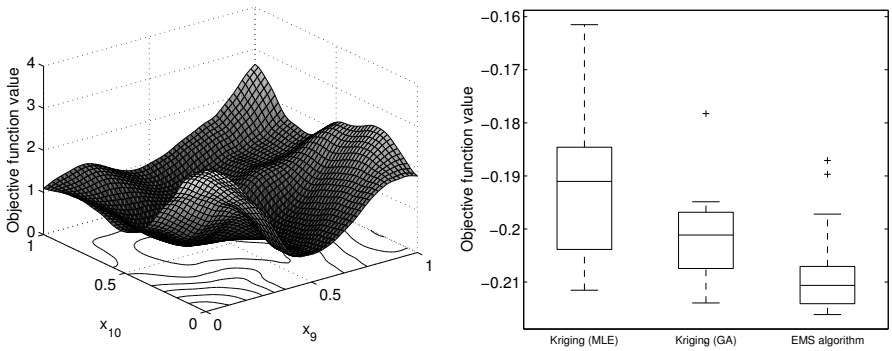


Figure 3.13.: a) 2D slice plot of the objective function. The y position of nodes 9 and 10 are set fixed to $y_9 = 0.6324$ and $y_{10} = 0.0975$, respectively. b) Box-and-whiskers plot of the final optimum.

out of 20 where a sole RBF model gives the best accuracy. Furthermore, the final ensemble models consist mostly of RBF models. The share of SVM models and Kriging models in those ensembles is identical.

3.2.7. Conclusion

This paper explored the use of evolutionary model selection (EMS) for surrogate-based optimization (SBO). The SBO framework of EGO is coupled with the EMS framework of Gorissen et al. [8] to solve an optimization problem from structural dynamics.

It is found that for a structural dynamics problem the EMS-based method outperforms traditional EGO using just Kriging models. Repeating the optimization process 20 times also shows using EMS results in less variance in the final optimum value than the other methods. Thus, these preliminary results are promising. Of course, using the EMS method comes at a higher computational cost as more surrogate model types are trained. The population of surrogate models of the EMS method is three times larger than Kriging (GA), as three different surrogate model types compete in the population.

Further testing on a wide range of benchmark problems is currently underway to see how the performance varies on other problems. The authors are also working on expanding the range of model types that can be included in the EMS algorithm. For example by using prediction variance estimation techniques for model types that do not support the prediction variance directly.

3.2.8. Bibliography

[1] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2006.

[2] D. Busby, C. L. Farmer, and A. Iske. Hierarchical nonlinear approximation for experimental design and statistical data fitting. *SIAM Journal on Scientific Computing*, 29(1):49–69, 2007.

[3] P-W. Chen, J-Y. Wang, and H-M. Lee. Model selection of SVMs using GA approach. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 3, pages 2035–2040, 25-29 July 2004.

[4] M. S. Eldred and D. M. Dunlavy. Formulations for surrogate-based optimization wiht data fit, multifidelity, and reduced-order models. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, Virginia*, pages AIAA–2006–7117, 2006.

[5] F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64:107–117, 2005.

[6] T. Goel, R. Haftka, W. Shyy, and N. Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33:199–216, 2007.

[7] D. Gorissen, K. Crombecq, I. Couckuyt, P. Demeester, and T. Dhaene. A surrogate modeling and adaptive sampling toolbox for computer based design. *Journal of Machine Learning Research*, 11:2051–2055, 2010.

[8] D. Gorissen, T. Dhaene, and F. DeTurck. Evolutionary model type selection for global surrogate modeling. *Journal of Machine Learning Research*, 10:2039–2078, 2009.

[9] A. Grosso, A.R.M.J.U. Jamali, and M. Locatelli. Finding maximin Latin hypercube designs by iterated local search heuristics. *European Journal of Operational Research*, 197(2):541–547, 2009.

[10] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Global Optimization*, 21:345–383, 2001.

[11] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[12] D.R. Jones, C.D. Perttunen, and B.E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Optimization Theory and Applications*, 79(1):157–181, 1993.

[13] A. J. Keane and A. P. Bright. Passive vibration control via unusual geometries: experiments on model aerospace structures. *Journal of Sound and Vibration*, 190(4):713–719, 1996.

[14] A. C. Keys, L. P. Rees, and A. G. Greenwood. Performance measures for selection of metamodels to be used in simulation optimization. *Decision Sciences*, 33:31–58, 2007.

[15] J. Knowles and H. Nakayama. Meta-modeling in multiobjective optimization. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 245–284. Springer-Verlag, Berlin, Heidelberg, 2008.

[16] S. Lessmann, R. Stahlbock, and S.F. Crone. Genetic algorithms for support vector machine model selection. In *Proceedings of the International Joint Conference on Neural Networks, 2006. IJCNN '06.*, pages 3063–3069, 16-21 July 2006.

[17] D. Lim, Y. Jin, Y. S. Ong, and B. Sendhoff. Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 14:329–355, 2008.

[18] D. Lim, Y-S. Ong, Y. Jin, and B. Sendhoff. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 07)*, pages 1288–1295, New York, NY, USA, 2007. ACM.

[19] J. Mockus, V. Tiesis, and A. Zilinskas. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

[20] Y-S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *American Institute of Aeronautics and Astronautics Journal*, 41(4):687–696, 2003.

[21] J. Peter, M. Marcelet, S. Burguburu, and V. Pediroda. Comparison of surrogate models for the actual global optimization of a 2d turbomachinery flow. In *Proceedings of the 7th WSEAS International Conference on Simulation, Modelling and Optimization*, pages 46–51, 2007.

[22] E. Sanchez, S. Pintos, and N.V. Queipo. Toward an optimal ensemble of kernel-based approximations with engineering applications. In *International Joint Conference on Neural Networks (IJCNN)*, volume 36, pages 247–261, 2006.

[23] András Sóbester, Stephen J. Leary, and Andy J. Keane. A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Structural and Multidisciplinary Optimization*, 27:371–383(13), 2004.

[24] András Sóbester, Stephen J. Leary, and Andy J. Keane. On the design of optimization strategies based on global response surface approximation models. *Global Optimization*, 33(1):31–59, 2005.

[25] D. P. Solomatine and A. Ostfeld. Data-driven modelling : some past experiences and new approaches. *Journal of hydroinformatics*, 10(1):3–22, 2008.

[26] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing Co., Pte, Ltd., Singapore, 2002.

[27] S. Tomioka, S. Nisiyama, and T. Enoto. Nonlinear least square regression by adaptive domain method with multiple genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 11(1):1–16, February 2007.

[28] F.A.C. Viana and R.T. Haftka. Why not run the efficient global optimization algorithm with multiple surrogates? In *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, pages AIAA 2010–3090, 2010.

[29] I. Voutchkov and A.J. Keane. Multiobjective Optimization using Surrogates. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture 2006. Proceedings of the Seventh International Conference*, pages 167–175, Bristol, UK, April 2006.

[30] X. Yao and Y. Xu. Recent advances in evolutionary computation. *Journal of Computer Science and Technology*, 21(1):1–18, 2006.

[31] C. Zhang, H. Shao, and Y. Li. Particle swarm optimisation for evolving artificial neural network. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 4, pages 2487–2490, 8-11 Oct. 2000.

[32] Z. Z. Zhou, Y. S. Ong, M. H. Lim, and B. S. Lee. Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Soft Computing*, 11(4):957–971, 2007.

[33] Z. Z. Zhou, Y-S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man and Cybernetics - Part C*, 37(1):66–76, 2007.

4

Multiobjective Surrogate-based Optimization

The saddest aspect of life right now is that science gathers knowledge faster than society gathers wisdom.
— Isaac Asimov

4.1. Fast calculation of Multiobjective Probability of Improvement and Expected Improvement criteria for Pareto Optimization

I. Couckuyt, D. Deschrijver, T. Dhaene

Submitted to the Journal of Global Optimization.

Abstract

The use of Surrogate-Based Optimization (SBO) is widely spread in engineering design to reduce the number of computational expensive simulations. However, "real-world" problems often consist of multiple, conflicting objectives leading to a set of competitive solutions (the Pareto front). The

objectives are often aggregated into a single cost function to reduce the computational cost, though a better approach is to use multiobjective optimization methods to directly identify a set of Pareto-optimal solutions, which can be used by the designer to make more efficient design decisions (instead of weighting and aggregating the costs upfront). Most of the work in multiobjective optimization is focused on MultiObjective Evolutionary Algorithms (MOEAs). While MOEAs are well-suited to handle large, intractable design spaces, they typically require thousands of expensive simulations, which is prohibitively expensive for the problems under study. Therefore, the use of surrogate models in multiobjective optimization, denoted as MultiObjective Surrogate-Based Optimization (MOSBO), may prove to be even more worthwhile than SBO methods to expedite the optimization of computational expensive systems. In this paper, the authors propose the Efficient Multiobjective Optimization (EMO) algorithm which uses Kriging models and multiobjective versions of the Probability of Improvement (PoI) and Expected Improvement (EI) criteria to identify the Pareto front with a minimal number of expensive simulations. The EMO algorithm is applied on multiple standard benchmark problems and compared against the well-known NSGA-II, SPEA2 and SMS-EMOA multiobjective optimization methods.

4.1.1. Introduction

Surrogate modeling techniques, also known as metamodeling, are becoming rapidly popular in the engineering community to speed up complex, computational expensive design problems [33, 20]. Surrogate models, or metamodels, are mathematical approximation models that mimic the behavior of computational expensive simulation codes such as mechanical or electrical finite element simulations, or computational fluid dynamic simulations. This paper deals with the use of surrogate models for expediting the optimization of time-consuming (black-box) problems of a deterministic nature, in contrast to stochastic simulation.

While several types of surrogate modeling uses can be distinguished, this work is concerned with the integration of surrogate models into the optimization process, often denoted by Surrogate-Based Optimization (SBO) or Metamodel-Assisted Optimization (MAO). SBO methods typically generate surrogate models on the fly that are only accurate in certain regions of the input space, e.g., around potentially optimal regions. The generated surrogate models can then be used to intelligently guide the optimization process to the global optimum.

The focus of this work is the global SBO method based on the Probability of Improvement (PoI) and Expected Improvement (EI), popularized by Jones et al. [23]. These "*statistical criteria*" guide the selection of new data points in such a way that the objective function is optimized, while minimizing the number of expensive simulations. The advantage of EI and

PoI is that, besides the prediction (mean), the uncertainty (variance) of the surrogate model is taken into account as well, providing a balance between exploration[1] and exploitation[2]. Most often EI or PoI is used in conjunction with the Kriging surrogate model (Gaussian processes) [25] which provides by construction a prediction of the mean as well as the variance, but other surrogate models are also possible, such as Radial Basis Functions (RBF), Support Vector Regression (SVR) [13], etc.

The single-objective SBO problem is well described in literature, however, most (if not all) "*real-world*" problems actually consists of multiple, conflicting objectives leading to a set of Pareto-optimal solutions. Often the objectives are aggregated into a single cost function, e.g., using a weighted sum, that can be optimized by standard optimization techniques. Subsequently, by repeating this process many times using varying starting conditions, e.g., different set of weights, several solutions on the Pareto front can be found. On the other hand, a multiobjective optimization method can optimize the different objective functions simultaneously, and try to find the Pareto front in just a single run. Examples of such methods are primarily the MultiObjective Evolutionary Algorithms (MOEAs), e.g., the "Non-dominated Sorting Genetic Algorithm II" (NSGA-II; [14]), the "Strength Pareto Evolutionary Algorithm 2" (SPEA2; [39]) and the "$\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm" (SMS-EMOA; [5]).

Unfortunately, MOEAs typically require a massive amount of function evaluations, which is infeasible for computational expensive simulators. Hence, it is vital to economize on the number of function evaluations, e.g., by using surrogate models. MultiObjective Surrogate-based Optimization (MOSBO) methods only appeared quite recently in literature. Most work is focused on integrating surrogate models in MOEAs [36]. Gaspar et al. [19] use neural networks to either approximate the fitness function or as a local approximation technique to generate search points more efficiently. Voutchkov et al. [31] apply the NSGA-II algorithm to Kriging models instead of the expensive simulator. For an overview of available techniques and approaches the reader is referred to [28, 37].

While the PoI and EI approach is well-developed and used for single-objective SBO, its use in MOSBO is not well spread. Single-objective versions of EI and PoI are utilized by Knowles et al. [26, 27] to solve MOSBO problems. This approach, known as ParEGO, uses Kriging and EI to optimize a weighted sum of objective functions. By randomizing the weights every iteration several solutions along the Pareto front can be identified. More recently, Keane [24] proposed multiobjective versions of PoI and Euclidean distance-based EI. At the same time Emmerich et al. [17] proposed the hypervolume-based EI criterion. Similarly to a weighted sum, the multi-objective versions of EI and PoI aggregate information from the surrogate

[1] Improving the overall accuracy of the surrogate model (space-filling).
[2] Enhancing the accuracy of the surrogate model solely in the region of the (current) optimum.

models into a single cost function, balancing between exploration[1] and exploitation[3]. Unfortunately, only formulae for two objective functions are given by Keane as the statistical criteria become rather cumbersome and complex for a higher number of objective functions. Similarly, while Emmerich et al. [16] describe formulae for an arbitrary number of dimensions for the hypervolume-based EI, the computation cost increases at least exponentially with the number of objectives and, hence, has only been applied to two objectives.

The key contribution of this paper is the Efficient Multiobjective Optimization (EMO) algorithm which is a much more efficient method of evaluating multiobjective versions of the PoI and EI criteria for multiobjective optimization problems. In fact, the problem at hand is similar to calculating the hypervolume (a Pareto set quality estimator) [40] as will be shown below and, hence, hypervolume algorithms can be adapted to aid in the evaluation of the statistical criteria. Moreover, a new statistical criterion is proposed, based on the hypervolume-based EI, which is significantly cheaper to compute while still delivering promising results.

For an overview of Kriging please recall Section 2.2.2.1 on page 2-39. In Section 4.1.2, an overview of the EMO algorithm is given, including general expressions for the PoI and several variants of EI. Subsequently, a fundamental part needed for the calculation of the statistical criteria is discussed in Section 4.1.2.4. Afterwards, in Section 4.1.3 the EMO algorithm is tested on several functions from the DTLZ benchmark suite [15]. Lastly, in Section 4.1.4 conclusions and future work are discussed.

4.1.2. Efficient Multiobjective Optimization (EMO)

4.1.2.1. Overview

A flow chart of the EMO algorithm is shown in Figure 4.1. First an initial set of points X is generated and evaluated on the expensive objective functions $f_j(\mathbf{x})$, for $j = 1 \dots m$. Each objective function $f_j(\mathbf{x})$ is then approximated by a Kriging model. Based on the Kriging models useful criteria can be constructed that help in identifying Pareto-optimal solutions. After selecting a new point it is evaluated on the expensive objective functions $f_j(\mathbf{x})$, the Kriging models are updated with this new information and this process is repeated in an iterative fashion until some stopping criterion is met.

Of particular interest are the Probability of Improvement (PoI) and Expected Improvement (EI) statistical criteria which are widely used for single-objective optimization [22, 10]. Hence, it may be useful to extend the concept of the PoI and EI directly to multiobjective optimization. Multiobjective versions of the PoI and EI are defined for an arbitrary number of objective functions in Sections 4.1.2.2 and 4.1.2.3.

[3] Improving or augmenting the Pareto front.

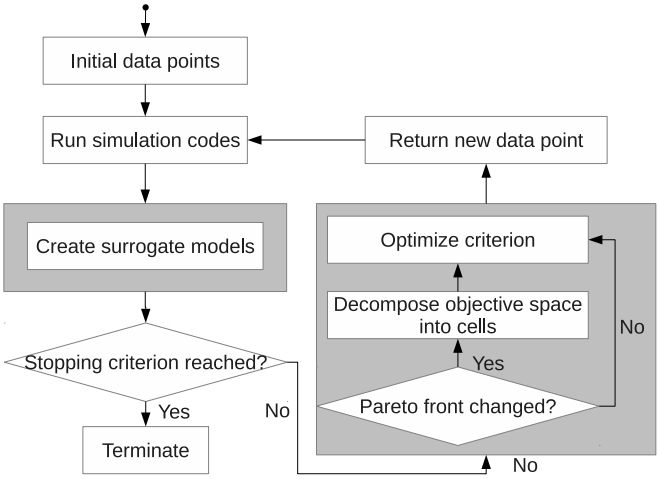


Figure 4.1.: Flow chart of the Efficient Multiobjective Optimization (EMO) algorithm.

For ease of notation in the forthcoming sections, the output of all the Kriging models can be considered as mutually independent Gaussian random variables $Y_j(\mathbf{x})$,

$$Y_j(\mathbf{x}) \sim \mathcal{N}(\mu_j(\mathbf{x}), s_j^2(\mathbf{x})) \; for \; j = 1 \ldots m. \tag{4.1}$$

The associated probability density function ϕ_j and cumulative distribution function Φ_j of $Y_j(\mathbf{x})$ are compactly denoted as,

$$\phi_j[y_j] \triangleq \phi_j[y_j; \mu_j(\mathbf{x}), s_j^2(\mathbf{x})], \tag{4.2}$$

$$\Phi_j[y_j] \triangleq \Phi_j[y_j; \mu_j(\mathbf{x}), s_j^2(\mathbf{x})]. \tag{4.3}$$

Consider a set of n samples, $(\mathbf{x}_1, \ldots, \mathbf{x}_n)^\top$ in d dimensions (see Equation 4.4) and associated function values, $\mathbf{y} = (y_1, \ldots, y_n)^\top$, where $(\cdot)^\top$ is the transpose of a vector or matrix.

$$X = \begin{pmatrix} \mathbf{x}_1, \ldots, \mathbf{x}_n \end{pmatrix}^\top = \begin{pmatrix} x_{1,1} & \ldots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \ldots & x_{n,d} \end{pmatrix} \tag{4.4}$$

Based on this set of points X, a Pareto set $\mathcal{P}$ can be constructed that comprises $v \leq n$ Pareto-optimal (non-dominated) solutions,

$$\mathcal{P} = \{\mathbf{f}(\mathbf{x}_1^*), \ldots, \mathbf{f}(\mathbf{x}_v^*)\}. \tag{4.5}$$

Each solution $\mathbf{f}(\mathbf{x}_i^*)$ is a vector that contains the objective function values for an associated input point $\mathbf{x}_i^* \in X$, for $i = 1 \dots v$,

$$\mathbf{f}(\mathbf{x}_i^*) = (f_1(\mathbf{x}_i^*), \dots, f_m(\mathbf{x}_i^*)) . \tag{4.6}$$

4.1.2.2. Probability of Improvement (PoI)

Evaluating the probability that the objective function values of a new input point $\mathbf{x}$ are located inside a well-defined region A in the objective space requires a multidimensional integration over that region. Naturally, several variants of the multiobjective PoI can be constructed as the concept of improvement is ambiguously defined in the context of multiobjective optimization. This is reflected in the selection of the integration region A, e.g., A can be the non-dominated part of the objective space or A can be the region in the objective space that solely extends the Pareto set, etc. In general, the probability that a new input point $\mathbf{x}$ yields improvement over the Pareto set $\mathcal{P}$ is denoted by the PoI $P[I]$,

$$P[I] = \int_{\mathbf{y} \in A} \prod_{j=1}^{m} \phi_j[y_j] dy_j. \tag{4.7}$$

To evaluate Equation (4.7), the integration area A can be decomposed into q (hyper-)rectangular cells, which yields a finite summation of contributing terms, see Figure 4.2a. The lower and upper bound $[\mathbf{l}^k, \mathbf{u}^k]$ of each cell, for $k = 1 \dots q$, will be computed in Section 4.1.2.4.

$$P[I] = \sum_{k=1}^{q} \pm \prod_{j=1}^{m} (\Phi_j[u_j^k] - \Phi_j[l_j^k]). \tag{4.8}$$

While the cells can be chosen to disjointedly cover the integration area A, the algorithm described in Section 4.1.2.4 decomposes the region A in overlapping cells. In this case, cells may negate the overlapping contribution of other cells by subtraction, denoted by the $\pm$ symbol in Equation (4.8).

4.1.2.3. Expected Improvement (EI)

While the PoI criterion is already quite useful and insensitive to the scaling of the objective functions, it does not, necessarily, encourage the generation of a uniform Pareto set. The EI quantifies the amount of improvement using an improvement function $I(\mathbf{y}, \mathcal{P})$ and, thus, prefers solutions that are lying farther from existing members of the Pareto set. The EI integral is defined as,

$$E[I] = \int_{\mathbf{y} \in A} I(\mathbf{y}, \mathcal{P}) \prod_{j=1}^{m} \phi_j[y_j] \, dy_j. \tag{4.9}$$

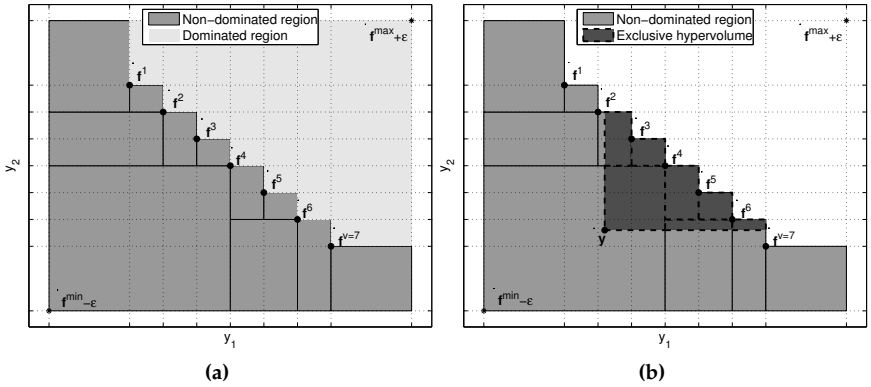


Figure 4.2.: Illustration of a Pareto set of two objective functions. The dots represent the Pareto points $\mathbf{f}^i$, for $i = 1 \dots v$, while $\mathbf{f}^{min}$ and $\mathbf{f}^{max}$ denote the ideal and anti-ideal point, respectively. a) The dark and light shaded regions denote the non-dominated and dominated region, respectively. The volume of the latter region is the hypervolume indicator, bounded by a reference point $\mathbf{r} = \mathbf{f}^{max} + \epsilon$. b) The integration area A of the hypervolume-based PoI corresponds to the (light and dark) shaded region which is decomposed into cells by a binary partitioning procedure. The exclusive hypervolume of a point $\mathbf{y}$ relative to the Pareto set can be computed from existing cells and corresponds to the dark shaded region.

In contrast to the PoI, it arguably makes more sense to only integrate the EI criteria over the region A corresponding to the non-dominated part of the objective space. The improvement function will automatically prefer new points that dominate the most points within the Pareto set $\mathcal{P}$ (the largest improvement). When no such points are found, the improvement function encourages the selection of points that extend the Pareto set $\mathcal{P}$ in an uniform way. Consequently, the design of the improvement function for the EI is crucial in identifying an optimal and uniform Pareto set. A good theoretical overview of different types of EI is given by [32], including work on scalar improvement functions [24, 16] as well as using the single-objective EI in a multiobjective setting [26, 21]. Below we focus on evaluating the Euclidean distance-based EI [24] as well as the hypervolume-based EI [17, 16] efficiently for many objectives. In addition, a simplified version of the hypervolume-based EI is proposed that is significantly cheaper to compute.

Hypervolume-based improvement function The hypervolume metric (or $\mathcal{S}$-metric) [40] is widely used in multiobjective optimization to assess the

quality of a Pareto set or to drive multiobjective optimization algorithms [5]. The hypervolume indicator $\mathcal{H}(\mathcal{P})$ denotes the volume of the region dominated by the Pareto set $\mathcal{P}$, bounded by a reference point $\mathbf{r}$ which needs to be dominated by all points of the Pareto set, see Figure 4.2a. Larger values of the hypervolume indicates better Pareto sets. Moreover, the exclusive hypervolume (or hypervolume contribution, see Figure 4.2b) of a Pareto set $\mathcal{P}$ relative to a point $\mathbf{p}$ is defined as,

$$\mathcal{H}_{exc}(\mathbf{p}, \mathcal{P}) = \mathcal{H}(\mathcal{P} \cup \{\mathbf{p}\}) - \mathcal{H}(\mathcal{P}). \tag{4.10}$$

$\mathcal{H}_{exc}$ measures the contribution (or improvement) of the point $\mathbf{p}$ to the Pareto set $\mathcal{P}$ and, hence, can also be used to define a scalar improvement function, namely,

$$I(\mathbf{p}, \mathcal{P}) = \begin{cases} \mathcal{H}_{exc}(\mathbf{p}, \mathcal{P}) & \textit{if } \mathbf{p} \textit{ is not dominated by } \mathcal{P} \\ 0 & \textit{otherwise} \end{cases}. \tag{4.11}$$

Subsequently, the integral of the hypervolume-based EI is,

$$E_{hv}[I] = \int_{\mathbf{y} \in A} I(\mathbf{y}, \mathcal{P}) \prod_{j=1}^{m} \phi_j[y_j]\, dy_j, \tag{4.12}$$

where A is the non-dominated region (bounded by the reference point $\mathbf{r}$).

Initially it was suggested to approximate the hypervolume-based EI using Monte Carlo techniques [17]. Recently, Emmerich et al. [16] proposed a method to calculate it exact for an arbitrary number of dimensions by decomposing the non-dominated region into a set of cells as is also done in this work. Unfortunately, in [16] the proposed mathematical expressions assume that the non-dominated region is decomposed into an uniform grid of cells based on the Pareto set, see the cells bounded by the dashed lines in Figure 4.2a. Hence, the number of cells required to evaluate the criterion scales at least exponentially with the number of Pareto points and objectives. Moreover, for each cell a separate hypervolume calculation needs to be done and, hence, it is infeasible to apply the method for three objectives or higher. This work develops a new mathematical expression for the hypervolume-based EI that alleviates some of its computational complexities by decomposing the non-dominated region into a much smaller set of cells as well as removing the separate hypervolume calculations.

In contrast to other statistical criteria, the proposed expressions for the hypervolume-based EI requires the non-dominated region to be covered by a set of *disjoint* cells, then the hypervolume-based EI can be written in closed form as,

$$E_{hv}[I] = \sum_{k=1}^{q} IC_k \tag{4.13}$$

where IC_k denotes the improvement contribution of cell k, namely,

$$\begin{aligned} IC_k &= \int_{l_1^k}^{u_1^k} \dots \int_{l_m^k}^{u_m^k} (\mathcal{H}(\mathcal{P} \cup \{\mathbf{y}\}) - \mathcal{H}(\mathcal{P})) \prod_{j=1}^{m} \phi_j[y_j] dy_j \\ &= \int_{l_1^k}^{u_1^k} \dots \int_{l_m^k}^{u_m^k} \sum_{k'=1}^{q} \prod_{j=1}^{m} (u_j^{k'} - max\{l_j^{k'}, l_j^k\}) \phi_j[y_j] dy_j \\ &= \sum_{k'=1}^{q} \prod_{j=1}^{m} \int_{l_j^k}^{u_j^k} (u_j^{k'} - max\{l_j^{k'}, l_j^k\}) \phi_j[y_j] dy_j \\ &= \sum_{k'=1}^{q} \prod_{j=1}^{m} G(l_j^k, u_j^k, l_j^{k'}, u_j^{k'}). \end{aligned}$$

For each cell $k' = 1 \dots q$ contributions are calculated per dimension as follows,

Type	$G(l_j, u_j, a_j, b_j) =$	Condition
A	$(b_j - a_j)(\Phi_j[\max(a_j, l_j)] - \Phi_j[l_j]) +$ $(b_j - y_j)(\Phi_j[\min(b_j, u_j)] - \Phi_j[\max(a_j, l_j)]) +$ $s_j^2(\mathbf{x})(\phi_j[\min(b_j, u_j)] - \phi_j[\max(a_j, l_j)])$	$b_j > l_j \wedge a_j < u_j$
B	$(b_j - a_j)(\Phi_j[u_j] - \Phi_j[l_j])$	$a_j \geq u_j$
C	0	otherwise

There are several ways a cell k' can contribute to the IC_k depending on its position relative to cell k, see Figure 4.3. Cell k is represented by the dark shaded cell. The pairs $(\cdot, \cdot)$ inside each cell $k' = 1 \dots q$ denote the type of contribution (A, B or C) per dimension. A, B and C refer to the Equations of the piecewise function G, see the previous Table. If a cell is completely covered by cell k in any one dimension, its length in that dimension is always included (type B). If a cell is only partially covered by the cell k, the contribution is divided into two parts: the integration when the cell is possibly fully covered (first line of A; zero if $a_j \leq l_j$) and when the cell is partially covered (second and third line of A). If a cell is not covered in any dimension, it also does not contribute to the IC_k (type C). This latter case, is not shown in Figure 4.3 but satisfies all cells that are to the left $(C, \cdot)$, bottom $(\cdot, C)$ or to the left and bottom (C, C) of cell k. The intermediary contribution IC_k is then obtained by multiplying the different kinds of contributions for each dimension and summing it over all cells $k' = 1 \dots q$.

The correctness of this algorithm has been verified by an extensive numerical comparison against the publicly available code of Emmerich et al.

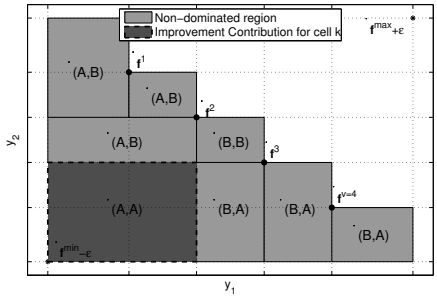


Figure 4.3.: The Improvement Contribution (IC) of the dark shaded cell k is calculated by multiplying different types of improvement (A, B or C) per dimension for all cells. The final IC is then obtained by summation over all cells.

for the two objective case. In addition, the two and three objective cases have been verified using Monte Carlo methods.

Regardless of the fact that the new procedure is already significantly cheaper than the method proposed by Emmerich et al., this hypervolume-based EI is still more expensive to evaluate than other statistical criteria. This is due to the computation time being more sensitive to the number of cells as well as the reliance on a binary partitioning of the non-dominated region into disjoint cells, which requires more cells to cover the integration area than the Walking Fish Group (WFG) algorithm explained in Section 4.1.2.4. Hence, a simplification of the hypervolume-based EI is proposed in the next section which can be evaluated using the WFG algorithm.

Hypervolume-based PoI Inspired by the definition of the hypervolume-based EI [16], the hypervolume-based PoI can be written as the product of the improvement function $I(\mu, \mathcal{P})$ and the PoI $P[I]$, and so the advantages of using the hypervolume contribution can be preserved while significantly reducing the overall computational complexity,

$$P_{hv}[I] = I(\mu, \mathcal{P}) \cdot P[I], \tag{4.14}$$

where $I(\mu, \mathcal{P})$ is defined as in (4.11) and $\mu = (\mu_1(\mathbf{x}), \ldots, \mu_m(\mathbf{x}))$ is a vector that contains the prediction of the Kriging models of each objective function for a point $\mathbf{x}$. In effect, the prediction variance is not taken into account anymore for the improvement function, in contrast to EI_{hv}, as it is moved outside of the integral.

The integration area A of $P[I]$ corresponds to the non-dominated region and, hence, a closed-form expression of the hypervolume-based PoI can be derived from the same set of cells used to evaluate $P[I]$, see Figure 4.2b, namely,

$$P_{hv}[I] = \sum_{j=1}^{q} \pm Vol(\mathbf{l}^j, \mathbf{u}^j) \cdot P[I] \tag{4.15}$$

where,

$$Vol(\mathbf{l}, \mathbf{u}) = \begin{cases} \prod_{i=1}^{m}(u_i - max(l_i, \mu_i)) & if\ u_i > \mu_i\ for\ i = 1 \dots m \\ 0 & otherwise \end{cases}.$$

Euclidean distance-based improvement function Similarly to the hypervolume-based PoI, Keane et al. [24] defines the EI as the product of the PoI $P[I]$ and an Euclidean distance-based improvement function. $E_{euclid}[I]$ for an input vector $\mathbf{x}$ is defined as,

$$E_{euclid}[I] = \sqrt{\sum_{i=1}^{m} w_i(\hat{\mathbf{y}}_i(\mathbf{x}) - \mathbf{f}_i^c)} \cdot P[I], \tag{4.16}$$

where $\hat{\mathbf{y}}(\mathbf{x})$ denotes the centroid of the $P[I]$ integral,

$$\hat{y}_i(\mathbf{x}) = \sum_{j=1}^{q} \hat{y}_i(\mathbf{x}; \mathbf{l}^j, \mathbf{b}^j), \tag{4.17}$$

with the centroid $\hat{\mathbf{y}}$ over arbitrary integral bounds $[\mathbf{l}, \mathbf{b}]$ defined by,

$$\begin{aligned} \hat{y}_i(\mathbf{x}; \mathbf{l}, \mathbf{b}) =& \textstyle\int_{l_1}^{u_1} \dots \int_{l_m}^{u_m} \phi_1[y_1] \dots y_i \phi[y_i] \dots \phi_m[y_m]\, dy_m \dots dy_1 / P[I] \\ =& \prod_{j=1}^{i-1}(\Phi_j[u_j] - \Phi_j[l_j]) \times \prod_{j=i+1}^{m}(\Phi_j[u_j] - \Phi_j[l_j]) \\ & \times \big(\mu_i(\mathbf{x})\Phi_i[u_i] - s_i^2(\mathbf{x})\phi_i[u_i] - \mu_i(\mathbf{x})\Phi_i[l_i] + s_i^2(\mathbf{x})\phi_i[l_i]\big) / P[I] \end{aligned} \tag{4.18}$$

Like all other EI criteria the integration area A of $P[I]$ and $\hat{\mathbf{y}}(\mathbf{x})$ is the non-dominated region. The weighted norm in (4.16) represents the Euclidean distance between the centroid $\hat{\mathbf{y}}(\mathbf{x})$ and the solution in $\mathcal{P}$ that is located closest to the centroid, i.e., $\mathbf{f}^c$,

$$\mathbf{f}^c = \underset{\mathbf{f}^c \in \mathcal{P}}{\operatorname{argmin}} \sqrt{\sum_{j=1}^{m} w_j(\hat{\mathbf{y}}_j(\mathbf{x}) - \mathbf{f}_j^c)}. \tag{4.19}$$

4.1.2.4. Decomposing the objective space into cells

In order to evaluate these statistical criteria efficiently, one or more integrals need to be evaluated over an integration area A. As A is non-rectangular and often irregularly shaped, especially for a higher number of objective

functions, the integral must first be decomposed into a sum of k integrals over rectangular cells. While these cells can be identified analytically upfront for two objectives [24] or one can use the most fine-grained cells possible (for a total of $q \approx (v+1)^m$ cells; [16]), this becomes rather prohibitively complex and cumbersome for a higher number of objective functions (> 2).

Instead, the authors propose to decompose the integration area in as few cells as possible using an efficient computer algorithm, i.e., each cell encompasses a large part of the integration area. A straightforward approach to determine the required bounds of the cells for the evaluation of the criteria is to use binary partitioning [11], see Figure 4.2a. While this approach is quite flexible as it allows to identify different kind of integration areas (e.g., leading to several variants of statistical criteria), it becomes prohibitively expensive as the number of objectives exceeds four. Nonetheless, by terminating the binary partitioning early the statistical criteria can still be approximated fairly well for a higher number of objectives.

However, the focus of this work is to improve the performance of the exact evaluation of the criteria. To that end, it makes sense to take advantage of the numerous algorithms for calculating the hypervolume. Formally, the hypervolume is the Lebesgue integral,

$$\mathcal{H}(\mathcal{P}) = \int_{\mathbf{y} \in A} 1 dy_1 \dots dy_m, . \tag{4.20}$$

where A is the region dominated by the Pareto set $\mathcal{P}$ and bounded by some reference point $\mathbf{r}$. As the statistical criteria are integrals evaluated over a similar area, but using a different integrand, the idea is to adapt a hypervolume routine and retrieve the integration area A as a set of cells instead of immediately calculating its (hyper)volume.

Exact algorithms [29, 38, 4, 3] for calculating the hypervolume as well as approximations [26, 7, 8, 18, 1, 2] and alternative versions of the hypervolume problem, e.g., finding the Pareto point(s) that contributes least to the hypervolume [9], have been suggested in literature. While the algorithm proposed by Beume et al. [3] has the best worst case complexity, the Walking Fish Group (WFG) algorithm [34] is actually faster on most practical optimization problems and, hence, is adapted in this work to evaluate the statistical criteria.

The basic WFG algorithm operates by defining the hypervolume as a sum of exclusive hypervolumes,

$$\mathcal{H}(\mathcal{P}) = \sum_{i=1}^{v} \mathcal{H}_{exc}(\mathbf{f}(\mathbf{x}_i^*); \mathcal{P} \backslash \{\mathbf{f}(\mathbf{x}_1^*), \dots, \mathbf{f}(\mathbf{x}_i^*)\}),$$

where each exclusive hypervolume in the summation corresponds to one of the slices bounded by dashed lines in Figure 4.4. At first sight this may look expensive as by definition the exclusive hypervolume (4.10) itself requires

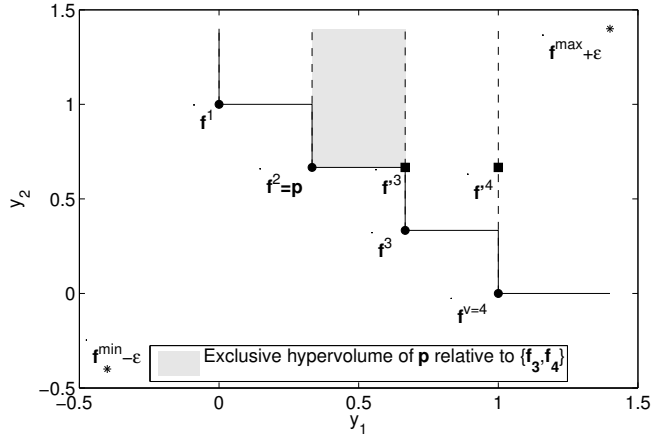


Figure 4.4.: The WFG algorithm calculates the hypervolume as a sum of exclusive hypervolumes, denoted by the cells (slices) bounded by the dashed lines. Each exclusive hypervolume slice is efficiently calculated as the volume of the cell bounded by $\mathbf{p}$ and $\mathbf{r}$ (in this case $\mathbf{r} = \mathbf{f}^{max} + \epsilon$) minus the hypervolume of a reduced Pareto set, represented by the squares, where all points are limited by the contributing point $\mathbf{p}$. This creates many dominating points (only one, $\mathbf{f}'_4$, in this illustration) which can be removed before continuing calculation.

two separate hypervolume calculations. Fortunately, several optimization can be made based on the following main ideas.

- Slices with smaller values than the contributing point $\mathbf{p}$ in any objective can be discarded as they contain no hypervolume that is dominated by $\mathbf{p}$.
- If $\mathbf{p}$ is dominated in the remaining objectives it dominates no more exclusive hypervolume.

Advantage of these insights can be taken by rewriting the exclusive hypervolume from (4.10) as $\mathcal{H}_{exc}(\mathbf{p}; \mathcal{P}) = \mathcal{H}(\{\mathbf{p}\}) - \mathcal{H}(\mathcal{P}')$, where $\mathcal{P}' = \{limit(\mathbf{p}, \mathbf{q}) | \mathbf{q} \in \mathcal{P}\}$ and $limit(\mathbf{p}, \mathbf{q}) = (\max\{p_1, q_1\}, \ldots, \max\{p_m, q_m\})$. The first term $\mathcal{H}(\{\mathbf{p}\})$ is simply the volume of the cell bounded by $\mathbf{p}$ and the reference point $\mathbf{r}$, the second term $\mathcal{H}(\mathcal{P}')$ is a recursion where the hypervolume is calculated for $\mathcal{P}'$. $\mathcal{P}'$ is obtained by taking the maximum ($limit$) of the contributing point and for each point in $\mathcal{P}$, i.e., the points are projected on $\mathbf{p}$, see Figure 4.4. Obviously, many points will be introduced that are dominated by some other point in $\mathcal{P}'$. These points do not contribute anything to the hypervolume and can be removed by taking the non-dominated subset of $\mathcal{P}'$ before continuing calculation. It is this last step

Algorithm 4.1 Pseudo-code of the adapted WFG algorithm with objective slicing. m is the number of objectives and **r** is the reference point.

```
1  signs = {1}
2  lb = {minpf}
3  ub = {r}
4
5  function adaptedWfg(P, m)
6    if P == {}
7      return;
8    end
9    sort P on objective m
10   return basecase2D(P) if m == 2
11
12   newSign = -signs(|signs|)
13   for each p in P
14         signs = signs U {newSign}
15     lb = lb U {p_1 ... p_m-1}
16     ub = ub U {r_1 ... r_m-1}
17
18     startCells = |signs|
19     adaptedWfg( nonDominatedSubset( limitSet( P, p ) )
           , m-1 );
20     endCells = |signs|
21
22     lb(startCells:endCells, m)= p_m
23     ub(startCells:endCells, m)= r_m
24   end
25 end
```

that significantly improves the performance as it has been shown that most datasets already lose over 50% of their points after only one recursion.

Note that if the contributing point **p** has generally higher objective values then likely more points can be pruned. Hence, it makes sense to process the worse Pareto points first by sorting the Pareto set $\mathcal{P}$ descending on the last objective before each iteration. Moreover, now the hypervolume calculation can also be sliced on the sorted objective, similarly to the Hypervolume by Slicing Objectives algorithm (HSO;[35]). A final optimization is the base case for two objectives, for which we can easily calculate the hypervolume in $O(v)$ assuming the Pareto set is sorted on the last objective.

The WFG algorithm is easily adapted to keep a record of a cell's lower- and upperbound instead of calculating its hypervolume. The adapted WFG algorithm, in case of minimization, identifies the cells that are dominated

by the Pareto set. Naturally, the algorithm can be modified to find the dominated region by temporarily viewing it as a maximization problem. However, for the evaluation of the statistical criteria, especially the EI, it is much more useful to identify the non-dominated region. This is achieved by subtracting the cells obtained from the adapted WFG algorithm from the cell that covers $\Re^m$ (possibly bounded by a reference point $\mathbf{r}$ when using one of the hypervolume-based criteria), see Figure 4.2a.

Hence, the adapted WFG algorithm can be used for statistical criteria where the integration area A corresponds to either the dominated or non-dominated region. Should other integration areas be required, e.g., for calculating the probability that a new point dominates at least two Pareto points, more flexible but slower methods such as binary partitioning [11] can be used. Pseudo-code of the full adapted WFG algorithm is found in Algorithm 4.1.

After $q << (v+1)^m$ sets of cells (=integral bounds) have been identified the actual PoI and EI statistical criteria can be evaluated using Equations (4.7), (4.17), (4.13) or (4.15). While evaluating the criteria the point $\mathbf{f}^{max} + \epsilon$ is replaced by $(\infty, \ldots, \infty)$ or $\mathbf{r}$ depending on the criterion.

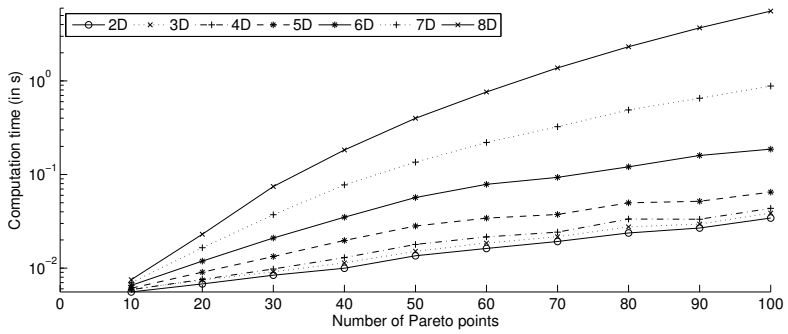
A plot with the practical computation time and the number of the cells is shown in Figures 4.5a and 4.5b, applying the adapted WFG algorithm to sets of Pareto points randomly drawn from the first quadrant of a unit sphere (taking the mean values of 1000 repetitions). The computation time of the cells poses no problem for the evaluation of the criteria (well within the seconds range). The limited factor of the EMO algorithm is the number of cells the integration area A is decomposed into, as for each cell the corresponding PoI or EI equations needs to be evaluated which can become prohibitively expensive when many cells are required to identify the integration area A.

The decomposition of the integration area A into cells and the actual evaluation of the PoI and EI criteria is separate in the sense that the cells only need to be identified once every sampling iteration, and then only if the (intermediate) Pareto set has changed with respect to the previous iteration. Afterwards, the PoI and EI criteria can be evaluated multiple times for a point $\mathbf{x}$, e.g., during optimization, using the same set of cells. Lastly, the hypervolume indicator is also easily obtained by summation of the volume of the cells.

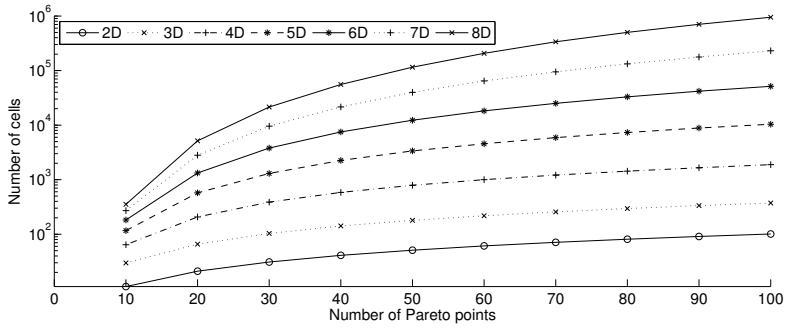
4.1.3. Examples

4.1.3.1. Introduction

A good set of configurable multiobjective benchmark problems has been proposed by Deb et al. [15], of which four benchmark functions are chosen and adapted slightly to benchmark the EMO algorithm. A summary of the selected benchmark functions is found in Table 4.1. For a complete



(a)



(b)

Figure 4.5.: a) Computation time of the integral bounds versus the number of Pareto points, for a different number of objective functions. The adapted WFG algorithm (C++ implementation) can easily be applied up to eight objectives. b) The number of cells versus the number of Pareto points. The evaluation of the statistical criteria is limited by the number of cells the integration area A is decomposed into.

Function	d	m	Reference point r
DTLZ1	6 inputs	3 objectives	$(400, 400, 400)$
DTLZ2	6 inputs	3 objectives	$(2.5, 2.5, 2.5)$
DTLZ7	6 inputs	4 objectives	$(1, 1, 1, 50)$
DTLZ5	6 inputs	6 objectives	$(2.5, 2.5, 2.5, 2.5, 2.5, 2.5)$

Table 4.1.: Summary of the DTLZ benchmark functions.

description of the benchmark functions the authors refer to [15].

All benchmark functions are configured to have six input parameters. Specifically, the first example is the DTLZ1 function with three objective functions where the Pareto front lies on the plane $y_1 + y_2 + y_3 = 1$. The second example is the DTLZ2 function with three objective functions where the Pareto front is the first quadrant of an unit sphere centered on the origin. The third example is the DTLZ7 function with four objective functions which has $2^{m-1} = 2^{4-1} = 8$ disconnected Pareto-optimal regions in the objective space. The last example, the DTLZ5 function configured to have six objective functions, is similar to DTLZ2 except that the Pareto front is just one slice of the unit hypersphere, i.e., the Pareto front is a (densely populated) curve in a $m = 6$ dimensional objective space.

4.1.3.2. Experimental setup

An initial set of 65 samples is generated by a near-optimal maximin Latin Hypercube Design (LHD; [12]). Subsequently, a statistical criterion is optimized each iteration to select the next point to evaluate. The criterion is optimized using a combination of Monte Carlo sampling and a local search. Specifically, $20 \times n$ Monte Carlo candidate points are generated and evaluated on the criterion. The best Monte Carlo candidate is further refined using Matlab's fmincon optimizer.

Various configurations of the EMO algorithm are applied on the benchmark functions. In particular, EMO is configured with the EI_{euclid} and P_{hv} criterion together with Kriging models using the Matérn correlation function [30] with $\nu = \frac{3}{2}$ and a constant regression function ($M = 1$ and $F = \mathbf{1}$). The hyperparameters of the Kriging models are optimized using SQPLab [6] (`http://www-rocq.inria.fr/~gilbert/modulopt/optimization-routines/sqplab/sqplab.html`), utilizing likelihood derivative information. The optional weight vector $\mathbf{w}$ of the EI_{euclid} criterion is set to $\mathbf{1}$ for all benchmark functions, except for DTLZ7 where $\mathbf{w} = (1, 1, 1, 0.02)$ to scale the last objective function into the same range as the other objective functions.

Furthermore, the EMO runs of the EI_{euclid} criterion are repeated with Kriging models using the Gaussian correlation function, these runs are

denoted by $EI_{euclid}^{\cdot}$ in the results. Lastly, extra EMO runs are configured for the DTLZ1 and DTLZ2 functions using the expensive EI_{hv} criterion with Kriging models using the Matérn correlation. Each of the in total 14 EMO runs is repeated 10 times for statistical robustness and halts when the sample budget is met, namely, 250 samples.

The EMO runs are compared against the NSGA-II, SPEA2 and SMS-EMOA evolutionary algorithms with a varying population size and maximum number of generations. The first run is configured with a population size of 25 and a maximum number of generations of 10 (total sample budget 250) and the second run is configured with a population size of 50 and a maximum number of generations of 50 (total sample budget 2500). The remaining parameters have been left to their default values. Similarly to the EMO runs, the evolutionary algorithm runs are repeated 10 times.

4.1.3.3. Results

Results for the benchmark functions have been summarized in Table 4.2. Note that the differences on the hypervolume metric are more significant than they appear because of the conservative choice of the reference point **r** (needed to accommodate the results of all test configurations).

Table 4.2.: Results of the EMO algorithm, NSGA-II, SPEA2 and SMS-EMOA. The best results for each test function are highlighted in bold, for each performance metric and within the same sample budget. The best results among the different configurations of the EMO algorithm are marked as italic.

Problem	|X|	Algorithm	Convergence measure		Hypervolume	
			Mean	Std	Mean	Std
DTLZ1	250	EI_{euclid}	93.2833	18.7840	6.3498e7	2.4970e5
		$EI_{euclid}^{\cdot}$	100.6741	14.2258	6.3650e7	1.2418e5
		EI_{hv}	***37.6112***	2.9315	*6.3940e7*	6.0452e4
		P_{hv}	66.9199	14.0029	6.3838e7	7.4330e4
		NSGA-II	75.8391	20.4219	6.3612e7	2.3441e5
		SPEA2	104.6259	0	6.3482e7	0
		SMS-EMOA	44.8818	7.9740	**6.3976e7**	8.0982e3
	2500	NSGA-II	16.6888	4.8071	6.3991e7	1.0227e4
		SPEA2	93.8381	0	6.3984e7	0
		SMS-EMOA	**9.5047**	2.8750	**6.4000e7**	324.0575

continued on next page

Problem	|X|	Algorithm	Convergence measure		Hypervolume	
			Mean	Std	Mean	Std
DTLZ2	250	EI_{euclid}	0.0843	0.0205	14.9423	0.0181
		$\dot{EI}_{euclid}$	0.1481	0.0133	14.8994	0.0114
		EI_{hv}	0.0411	0.0052	14.8834	0.0165
		P_{hv}	***0.0106***	0.0021	***15.0326***	0.0054
		NSGA-II	0.2725	0.0460	13.6238	0.2725
		SPEA2	0.1643	0	14.4873	0
		SMS-EMOA	0.0388	0.0071	14.9021	0.0160
	2500	NSGA-II	0.1497	0.0185	14.6435	0.0460
		SPEA2	0.1544	0.0298	14.8503	0
		SMS-EMOA	**0.0030**	2.8954e-4	**15.0280**	3.4727e-4

continued on next page

Problem	|X|	Algorithm	Convergence measure		Hypervolume	
			Mean	Std	Mean	Std
DTLZ7	250	EI_{euclid}	4.3888	2.8159	42.4629	0.4042
		$\dot{EI}_{euclid}$	1.7066	1.4069	42.6332	0.3295
		P_{hv}	***0.0280***	0.0037	***43.5404***	0.0188
		NSGA-II	13.9371	2.3112	23.2392	5.4733
		SPEA2	10.1169	0	37.4830	0
		SMS-EMOA	3.4186	2.2457	41.2087	1.6529
	2500	NSGA-II	9.6799	2.3516	30.7966	4.2005
		SPEA2	5.4330	0	42.1191	0
		SMS-EMOA	**0.0236**	0.0015	**43.7127**	0.0953
DTLZ5	250	EI_{euclid}	0.2259	0.0019	197.1390	0.1453
		$\dot{EI}_{euclid}$	0.2286	0.0013	196.8852	0.1777
		P_{hv}	*0.0835*	0.0053	***198.6425***	0.1563
		NSGA-II	0.0656	0.0376	192.1285	2.0064
		SPEA2	0.1475	0	192.6617	0
		SMS-EMOA	**0.0467**	0.0268	196.0038	0.6004
	2500	NSGA-II	**0.0727**	0.0162	194.9017	0.3805
		SPEA2	0.2151	0	194.3750	0
		SMS-EMOA	0.1141	0.0070	**198.5351**	0.0343

In general, it is seen that the EMO runs have better performance than the MOEAs in terms of hypervolume score for most functions except for DTLZ1. After a closer examination it is observed that the accuracy of the Kriging models of DTLZ1 for most statistical criteria is sub-optimal. In particular, the first objective function is difficult to approximate using the Kriging models, see Figure 4.6a.

A plot of the final Pareto sets generated by the EI_{euclid} and P_{hv} runs for the DTLZ2 problem is shown in Figure 4.7. It is seen that the hypervolume-based criterion emphasizes the edges of Pareto front more while leaving a small gap between the edge and the inner portion of the Pareto front. This is not unlike the DTLZ2 results as reported in [5] and is due to the nature of the hypervolume indicator. Logically, the farther away the reference point is located, the larger the exclusive hypervolume will be for points lying on the edge of the current Pareto set (as the exclusive hypervolume is then solely bounded by the reference point). Further research is needed to determine the influence of the choice of reference point $\mathbf{r}$ on the statistical criteria [1].

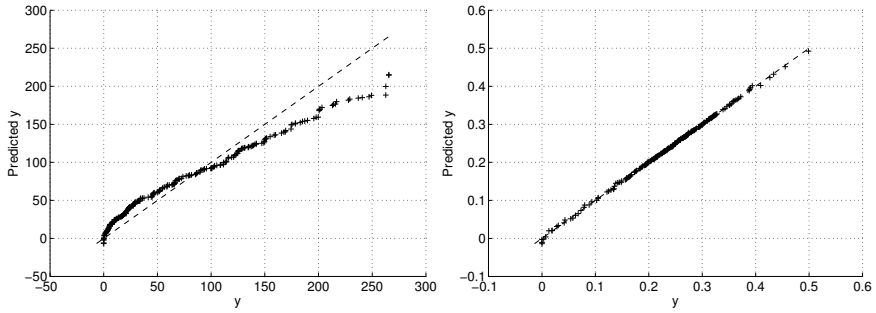


Figure 4.6.: 20-fold cross validation applied on the Kriging models based on 250 samples. The black dots denote the cross validated prediction values versus the real objective values. a) Final Kriging model of the first objective function of the DTLZ1 function. It is seen that Kriging has problems approximating the upper portion of the objective function. b) Final Kriging model of the first objective function of the DTLZ5 function. Kriging is able to approximate the objective function quite well.

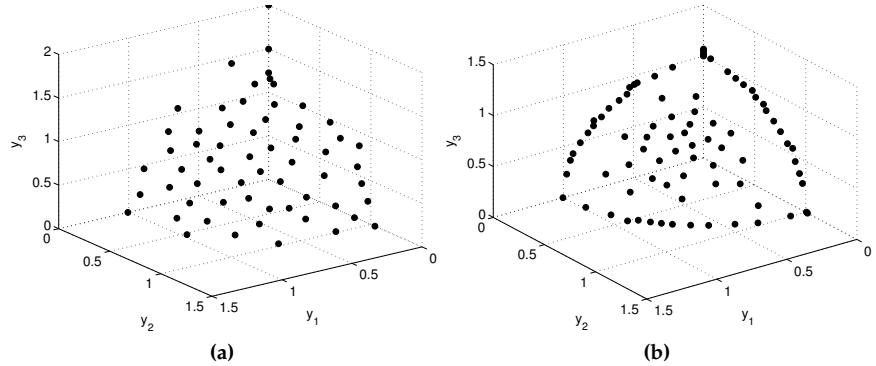


Figure 4.7.: Generated Pareto sets of the a) EI_{euclid} and b) P_{hv} EMO runs for the DTLZ2 function. The hypervolume-based metric focuses more on sampling the edge (extrema) of the Pareto front, while the Euclidean distance-based criterion performs a seemingly more uniform search over the Pareto front, though it performs slightly worse on the hypervolume metric.

Lastly, a matrix of contour plots of the P_{hv} criterion based on the final Kriging models of the DTLZ7 function can be found in Figure 4.8. Each contour plot is in function of the x_1 and x_2 parameters while the remaining parameters are set fixed. In particular, the parameters x_3 and x_5 are varied in three discrete steps along the columns of the matrix. Similarly, the parameters x_4 and x_6 are varied along the rows. The samples are denoted by black dots, the larger a dot the closer the sample lies to the actual slice. To avoid clutter, samples lying further than 0.25 from the slice are not shown. It is seen that the region $x_4 = x_5 = x_6 = 0$, corresponding to the location of the real Pareto front, is densely sampled by the criterion, while not completely neglecting other regions of the input space.

While the EMO algorithm outperforms the MOEAs on the hypervolume indicator on most problems, there are some limitations. The EMO algorithm, and other MOSBO techniques, rely on the quality of the surrogate model to guide the selection of new expensive data points. While the Kriging models do not have to be accurate at the start of the algorithm when using the EI and PoI criteria, the Kriging models should be able to capture the behavior of the objective functions sufficiently well when enough samples become available, which might not always be the case (see Figure 4.6 and the DTLZ1 results). Furthermore, the construction of the Kriging models and the evaluation of the statistical criteria comes at a computational cost, similar to the computational cost of MOEAs that rely on the hypervolume, which might limit the practical usage of the EMO algorithm for some (less expensive) optimization problems.

4.1.4. Conclusion

The authors presented the Efficient Multiobjective Optimization (EMO) algorithm, which uses multiobjective versions of the Probability of Improvement (PoI) and Expected Improvement (EI) to identify the Pareto front with a limited sample budget. Different configurations of the EMO algorithm are compared against the well-known SPEA2, NSGA-II and SMS-EMOA evolutionary methods with promising results. In theory an arbitrary number of objective functions can be handled. However, in practice due to the nature of the multiobjective EI and PoI statistical criteria EMO also does not escape the curse of dimensionality (no-free-lunch theorem) with respect to the number of objective functions and number of Pareto points. Nevertheless, the EMO algorithm can be applied to problems up to eight objectives.

Future work will focus on minimizing the number of cells and on an iterative update scheme for the cells, which will be considerable more efficient than recalculating the cells almost each iteration. Indirectly, a speedup can also be achieved by selecting multiple update points at a time. Finally, it may also be worthwhile to investigate the use of approximated statistical criteria, namely, adapting hypervolume approximation routines for decomposing the integration area into cells.

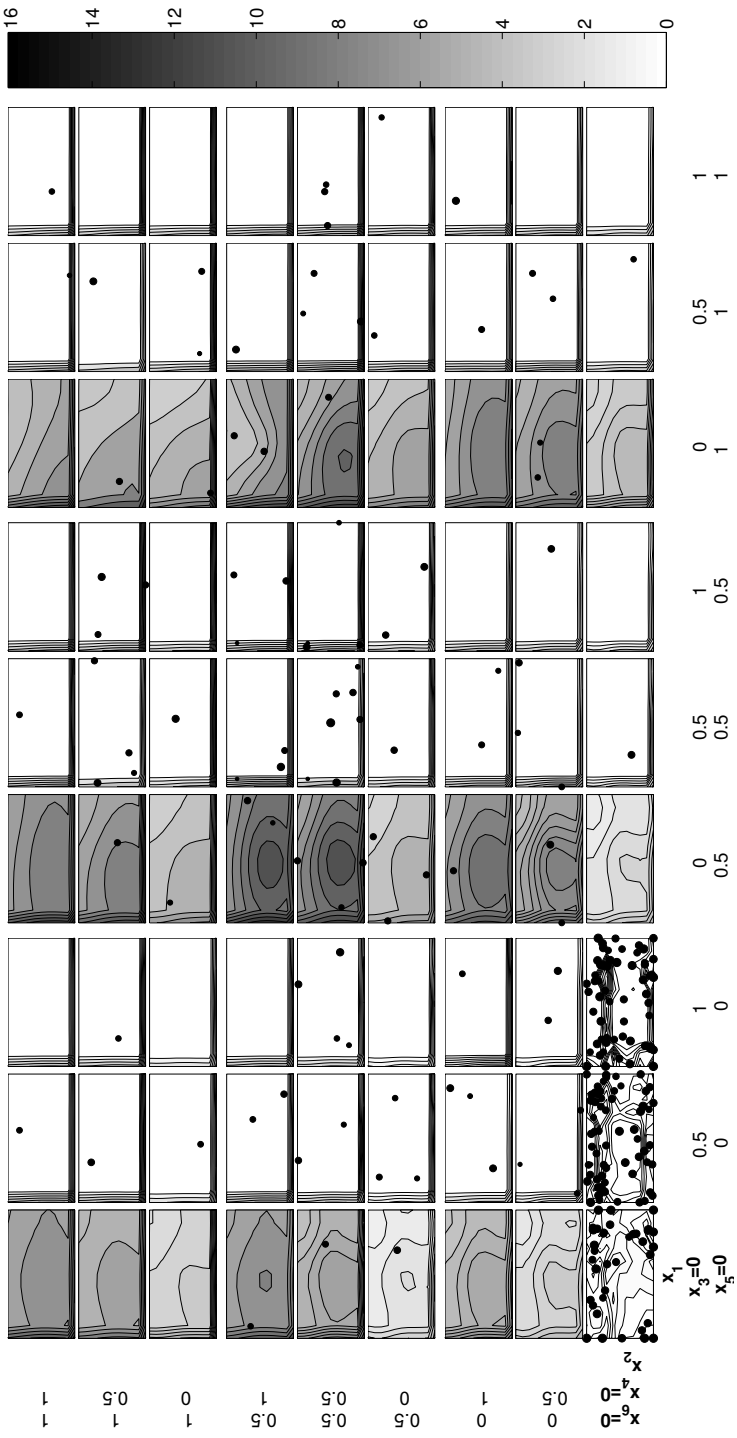


Figure 4.8.: Contour plots arranged in a matrix of the P_{hv} criterion based on the final Kriging models of the DTLZ7 function. The samples are denoted by black dots, the larger a dot the closer the sample lies to the actual slice. The real Pareto front is located at $x_4 = x_5 = x_6 = 0$.

4.1.5. Bibliography

[1] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point. In *Workshop Foundation Genetic Algorithms*, 2009.

[2] J. Bader, K. Deb, and E. Zitzler. Faster hypervolume-based search using monte carlo sampling. In *Multiple Criteria Decision Making Sustainable Energy Transportation System*, 2010.

[3] N. Beume. S-metric calculation by considering dominated hypervolume as klee's measure problem. *Evolutionary Computation*, 17(4):477–492, 2009.

[4] N. Beume, C. M. Fonseca, and M. López-Ibá nez. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, 2009.

[5] N. Beume, B. Naujoks, and M.T.M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[6] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2006.

[7] K. Bringmann and T. Friedrich. Approximating the least hypervolume contributor: Np-hard in general, but fast in practice. In *Evolutionary Multiobjective Optimization*, 2009.

[8] K. Bringmann and T. Friedrich. Don't be greedy when calculating hypervolume contributions. In *Workshop Foundation Genetic Algorithms*, 2009.

[9] K. Bringmann and T. Friedrich. An efficient algorithm for computing hypervolume contributions. *Evolutionary Computation*, 18(3):383–402, 2010.

[10] I. Couckuyt, F. Declercq, T. Dhaene, and H. Rogier. Surrogate-based infill optimization applied to electromagnetic problems. *Journal of RF and Microwave Computer-Aided Engineering: Advances in design optimization of microwave/rf circuits and systems*, 20(5):492–501, 2010.

[11] I. Couckuyt, D. Deschrijver, and T. Dhaene. Towards efficient multiobjective optimization: Multiobjective statistical criterions. In *IEEE World Congress on Computational Intelligence*, pages 1–7, 2012.

[12] E.R. Dam, B.G.M. van Husslage, D. den Hertog, and J.B.M. Melissen. Maximin Latin hypercube designs in two dimensions. *Operations Research*, 55(1):158–169, 2007.

[13] Jos de Brabanter. *LS-SVM Regression Modelling and its Applications*. PhD thesis, Katholieke Universiteit Leuven, 2004.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[15] K. Deb, L. Thiele, M. Laummans, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical Report 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

[16] M.T.M. Emmerich, A.H. Deutz, and J.W. Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *IEEE Congress on Evolutionary Computation (CEC)*, 2011.

[17] M.T.M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[18] T. Friedrich, C. Horoba, and F. Neumann. Multiplicative approximations and the hypervolume indicator. In *Genetic Evolutionary Computation*, 2009.

[19] A. Gaspar-Cunha and A. Vieira. A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. *International Journal of Computers, Systems, and Signals*, 6(1):18–36, 2004.

[20] D. Gorissen, K. Crombecq, I. Couckuyt, P. Demeester, and T. Dhaene. A surrogate modeling and adaptive sampling toolbox for computer based design. *Journal of Machine Learning Research*, 11:2051–2055, 2010.

[21] S. Jeong and S. Obayashi. Efficient global optimization (ego) for multi-objective problem and data mining. In *Congress on Evolutionary Computation*, 2010.

[22] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Global Optimization*, 21:345–383, 2001.

[23] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[24] A. J. Keane. Statistical improvement criteria for use in multiobjective design optimization. *AIAA Journal*, 44(4):879–891, 2006.

[25] J. P.C. Kleijnen. *DASE : Design and Analysis of Simulation Experiments*. Springer, May 2007.

[26] J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[27] J. Knowles, D. Corne, and A. Reynolds. Noisy multiobjective optimization on a budget of 250 evaluations. In *5th International Conference on Evolutionary Multi-Criterion Optimization*, 2009.

[28] J. Knowles and H. Nakayama. Meta-modeling in multiobjective optimization. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 245–284. Springer-Verlag, Berlin, Heidelberg, 2008.

[29] J. D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, University of Reading, UK, 2002.

[30] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[31] I. Voutchkov and A.J. Keane. Multiobjective Optimization using Surrogates. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture 2006. Proceedings of the Seventh International Conference*, pages 167–175, Bristol, UK, April 2006.

[32] T. Wagner, M. Emmerich, A. Deutz, and W. Ponweiser. On expected-improvement criteria for model-based multi-objective optimization. *Parallel Problem Solving From Nature*, 6238:718–727, 2010.

[33] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[34] L. While, L. Bradstreet, and L. Barone. A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16:86–95, 2012.

[35] L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *Evolutionary Computation*, 10(1):29–38, 2006.

[36] Q. Zhang, W. Liu, E. Tsang, and B. Virginas. Expensive multiobjective optimization by moea/d with gaussian process model. *Evolutionary Computation*, 14(3):456–474, 2010.

[37] A. Zhou, B-Y Qu, H. Li, S-Z Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1:32–49, 2011.

[38] E. Zitzler. Hypervolume metric calculation. Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2001.

[39] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the performance of the strength pareto evolutionary algorithm. Technical report, Swiss Federal Institute of Technology, 2001.

[40] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V. Grunert da Fonseca. Performance assesment of multiobjective optimizers: an analysis and review. *Evolutionary Computation*, 7(2):117–132, 2003.

5 Inverse surrogate modeling

Without music, life would be a mistake.
— Friedrich Nietzsche, Twilight of the Idols

5.1. Identification of quasi-optimal regions in the design space using surrogate modeling

I. Couckuyt, J. Aernouts, D. Deschrijver, F. De Turck, T. Dhaene

Published in Engineering with Computers,

January 2012.

Abstract

The use of Surrogate-Based Optimization (SBO) is widely spread in engineering design to find optimal performance characteristics of expensive simulations (forward analysis: from input to optimal output). However, often the practitioner knows a priori the desired performance and is interested in finding the associated input parameters (reverse analysis: from desired output to input). A popular method to solve such reverse (inverse) problems is to minimize the error between the simulated performance and the desired goal. However, there might be multiple quasi-optimal solutions to

the problem. In this paper, the authors propose a novel method to efficiently solve inverse problems and to sample Quasi-Optimal Regions (QORs) in the input (design) space more densely. The development of this technique, based on the probability of improvement criterion and Kriging models, is driven by a real-life problem from bio-mechanics, i.e., determining the elasticity of the (rabbit) tympanic membrane, a membrane that converts acoustic sound wave into vibrations of the middle ear ossicular bones.

5.1.1. Introduction

This paper is concerned with efficiently solving complex, computational expensive design problems using surrogate modeling techniques [25]. Surrogate models, also known as metamodels, are cheap approximation models for computational expensive (black-box) simulations. Surrogate modeling techniques are well-suited to handle, for example, expensive finite element (FE) simulations, computational fluid dynamic (CFD) simulations and, of course, physical experiments. In particular, the research in this paper is concerned with deterministic computer codes, in contrast to non-deterministic (stochastic) problems.

Depending on the construction and usage of surrogate models several modeling flavors can be distinguished. Surrogate models can be built upfront to approximate the simulation code accurately over the entire input (design) space and, hence, can afterwards be used to replace the expensive code for design, analysis and optimization purposes. On the other hand, the construction of surrogate models can also be integrated in the optimization process. Usually, the latter case, known as Surrogate-Based Optimization (SBO), generates surrogate models on the fly that are only accurate in certain regions of the input space, e.g., around optimal regions.

The construction of surrogate models as efficiently as possible is an entire research domain in itself. In order to come to an acceptable model, numerous problems and design choices need to be overcome (what data collection strategy to use, which variables are relevant, how to integrate domain knowledge, etc.). Other aspects of surrogate modeling include choosing the right type of approximation model for the problem at hand, a tuning strategy for the surrogate model parameters (=hyperparameters), and a performance measure to asses the accuracy of the surrogate model [13].

The general work-flow of surrogate modeling is illustrated in Figure 5.1. First, an experimental design, e.g., from Design of Experiments (DoE), is specified and evaluated. Subsequently, surrogate models are built to fit this data as well as possible, according to a set of measures (e.g., cross validation). The hyperparameters are estimated using an optimization algorithm. The accuracy of the set of surrogate models is improved until no further improvement can be obtained (or when another stopping criterion, such as a time limit, is met). If the stopping criteria are satisfied the process

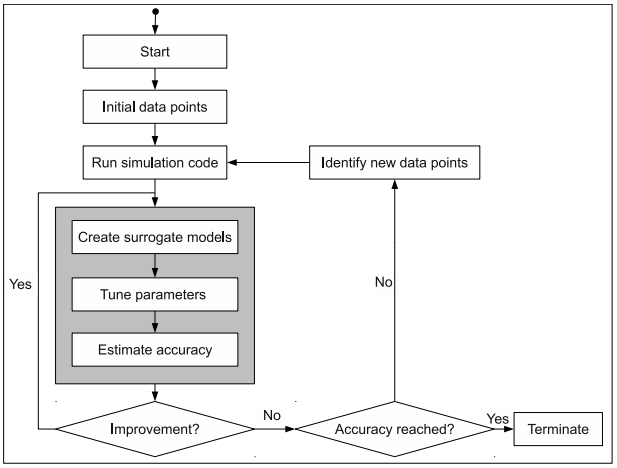


Figure 5.1.: Flow chart of the surrogate modeling process [13].

is halted and the final, best surrogate model is returned. On the other hand, when no stopping criterion is met, a sequential design strategy, also known as active learning or adaptive sampling, will select new data points to be evaluated and the surrogate models are updated with this new data.

Most often, surrogate models are used to solve so-called "*forward problems*". The practitioner is interested in the output or performance characteristics of the simulation system given the input (design) parameters. The surrogate models define the mapping between the input space (design space) and the output space (performance space). Examples of forward problems are found in validation and verification, sensitivity analysis, and optimization.

In contrast, the focus of the "*reverse (inverse) problem*" is on exploring the input space. Ideally, a surrogate model could be created that maps the output parameters to the input parameters (as opposite to forward modeling) of the complex system over the entire output space. However, many inverse problems are ill-posed. Considering Hadamard's definition of ill-posedness [14], the two outstanding problems hampering the creation of a full inverse surrogate model are non-uniqueness and instability. A good overview of the associated intricacies is presented by Barton in [3]. For all the above reasons, the inverse problem is often reduced to the task of finding one (or more) input parameter combination for a certain output characteristic. Still, it is possible that,

1. no such input parameter combination exists,

2. more than one input parameter combination satisfies the given output characteristic(s).

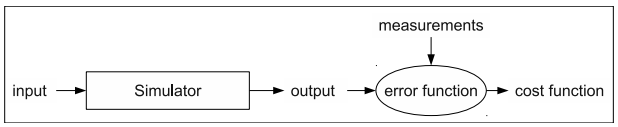


Figure 5.2.: The inverse problem is often solved by minimizing the error function between the simulation output and the measured data.

A typical inverse problem is the estimation of some (physical) material or design parameter, e.g., the permittivity of a substrate [5] or the elasticity of rubber [1], given the desired output or system behavior. A popular solution is to convert the reverse problem to a (forward) optimization problem. Namely, a simulation model is constructed, parametrized by the properties or design parameters of interest. By minimizing the error between the parametrized simulation model and the measured data the input parameters (material properties) of the simulation model are obtained that correspond with the measurements or desired output, see Figure 5.2.

The focus of this paper is to efficiently solve inverse problems where an infinite number of input parameter combinations is possible, i.e., whole regions in the input space that satisfy the cost function sufficiently well. From now on these regions in the input space will be denoted Quasi-Optimal Regions (QORs). Hence, traditional (surrogate-based) optimization of the cost function is insufficient.

Recently, Picheny et al. [18] presented a scheme to sample the input regions that correspond to an output region of interest. While using a similar approach, our work focuses on finding QORs as efficiently as possible. Moreover, the approach of Picheny et al. requires expensive numerical integration and the Kriging model must be updated for every new sample point (though alternative, faster approaches are discussed). We present in this work a simple and cheap criterion in combination with a space-filling criterion to ensure proper coverage of the input domain.

The main contribution of this paper is a novel sequential design strategy, denoted as the "*QOR sampling algorithm*", that is able to efficiently sample QORs densely, in a quasi-uniform way. The surrogate model of choice is the Gaussian Process (GP) based Kriging. Kriging is a popular surrogate model for the approximation of deterministic computer code [20]. GPs enable the use of statistical infill criteria in a sequential design strategy. The presented method consists of the extension of such a statistical infill criterion, namely, the Probability of Improvement (PoI) [15], and a new search strategy to exploit the infill criterion.

The QOR sampling algorithm has been implemented in a flexible research platform for surrogate modeling, the **SU**rrogate **MO**deling (SUMO) Toolbox [13] (The SUMO Toolbox can be downloaded from: `http://sumo.intec.ugent.be`. An AGPL open source license is available for research purposes)

and has been applied to a real-life problem from bio-mechanics, i.e., determining the elasticity of the (rabbit) tympanic membrane, a membrane that converts acoustic sound wave into vibrations of the middle ear ossicular bones.

Section 5.1.2 describes the use of infill criteria and, in particular, describes an extension of PoI needed to solve the engineering problem at hand. A new search strategy to exploit infill criteria is described in Section 5.1.3. The QOR sampling algorithm is applied to the Branin and Hartman functions in, respectively, Sections 5.1.4 and 5.1.5. The engineering problem from bio-mechanics is presented in Section 5.1.6. Details of the SUMO Toolbox configuration are found in Section 5.1.6.2. Results of the engineering problem and conclusions form the last two sections of this paper, i.e., Sections 5.1.6.3 and 5.1.7.

5.1.2. Infill criteria

In engineering, infill criteria are (sampling) functions, also known as figures of merit or metrics, that measure how *interesting* a data point is in the input space. Starting from an initial approximation of the simulation system, new sample points (infill or update points) are selected based on an infill criterion. The scope of infill criteria ranges from increasing the accuracy of the prediction (e.g., for creating globally accurate surrogate models) to the prediction itself to facilitate optimization. In global SBO it is crucial that the infill criterion is a balance between exploration (enhancing the overall accuracy of the surrogate model) and exploitation (enhancing the accuracy of the surrogate model solely in the region of the (current) optimum).

A well-known infill criterion that is able to effectively solve this trade-off is Expected Improvement (EI), which has been popularized by Jones et al. [16, 22, 11] as the Efficient Global Optimization (EGO) algorithm. Jones wrote an excellent discussion regarding the infill criteria approach in [15]. Subsequently, Sasena compared different infill criteria for optimization and investigated extensions of those infill criteria for constrained optimization problems in [21].

5.1.2.1. Generalized Probability of Improvement (gPoI)

Among several statistical infill criteria investigated by Jones the Probability of Improvement (PoI), see Section 3.1.2.2 on page 3-5, is used and generalized in this work. While PoI is a very useful infill criterion for optimization, it only focuses on the global optimum, not on a range of output values. The authors extend the idea of the PoI criterion to allow identification of an arbitrarily band in the output space. Let $[T_1, T_2]$ be the range of interest in the output space. The generalized Probability of Improvement (gPoI) is defined as the probability that the function value $f(\mathbf{x})$ at a point $\mathbf{x}$ lies with the output range $[T_1, T_2]$,

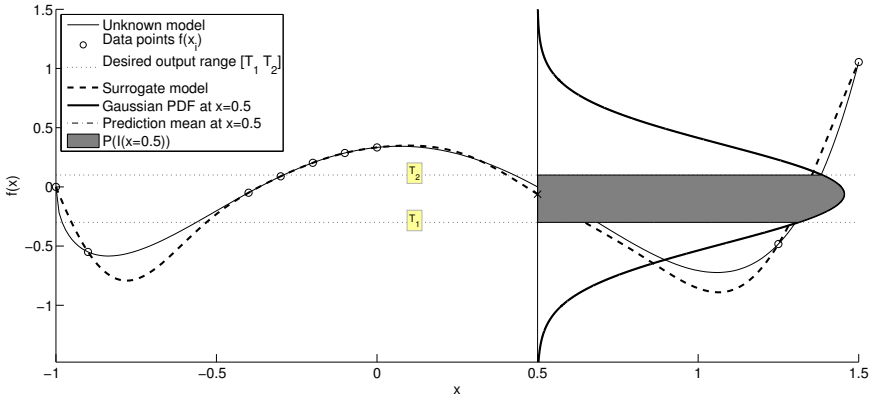


Figure 5.3.: Graphical illustration of a Gaussian Process (GP) and the generalized Probability of Improvement (gPoI). A surrogate model (dashed line) is constructed based on some data points (circles). For each point the surrogate model predicts a Gaussian probability density function (PDF). E.g., at $x = 0.5$ an example of such a PDF is drawn. The volume of the shaded area is the gPoI based on the desired output range $[T_1, T_2]$.

$$\begin{aligned} gPoI(\mathbf{x}) = P(T_1 \leq Y(\mathbf{x}) \leq T_2) &= \int_{T_1}^{T_2} Y(\mathbf{x})\, dY \\ &= P(Y(\mathbf{x}) \leq T_2) - P(Y(\mathbf{x}) \leq T_1) \\ &= \int_{-\infty}^{T_2} Y(\mathbf{x})\, dY - \int_{-\infty}^{T_1} Y(\mathbf{x})\, dY \\ &= \Phi\left(\frac{T_2 - \hat{y}}{\hat{s}}\right) - \Phi\left(\frac{T_1 - \hat{y}}{\hat{s}}\right), \end{aligned} \tag{5.1}$$

where $\Phi(t)$ is the standard normal cumulative distribution function $\Phi(t) = \frac{1}{2}\left[1 + erf\left(\frac{t}{\sqrt{2}}\right)\right]$ and $erf(\cdot)$ is the error function. Note that the standard abbreviation "PoI" is not well-suited anymore as the focus is now on sampling an interval instead of improving the optimum.

The gPoI criterion has recently been successfully applied to quasi-uniformly sample the region(s) in the input space that correspond to a desired interval $[T_1, T_2]$ in the output space [6] (T_1 and T_2 are defined upfront). In essence, input (design) parameters are sought that correspond with a certain set of performances (= inverse problem).

In this paper, one wants to find the QORs which include all near-optimal solutions. So, the lower bound can be defined as $T_1 = -\infty$ and T_2 is defined on the fly. By varying and tightening the upper bound T_2 of the integral (see Eq. 5.1) dynamically during the optimization process the QOR can be accurately identified.

The authors suggests to use the intermediate minimal function value $\hat{f}_{min}$ plus a percentage p of $|\hat{f}_{min}|$ as upper bound, namely, $T_2 = \hat{f}_{min} + p \cdot |\hat{f}_{min}|$. Thus, all input parameter combinations are sought that lie within a desired percentage p of $\hat{f}_{min}$.

Furthermore noise can be taking into account by adding an extra offset to the upper bound. This might be required as many inverse problems involve noisy measurements, as will be shown in the application of Section 5.1.6. To that end, regression Kriging [24] is used, as explained in Section 2.2.2.4, where a parameter 10^{λ} gives an indication of the amount of noise. Furthermore, as λ is determined during the MLE of Kriging an estimate of the noise variance τ^2 can be calculated as $\hat{\tau}^2 = 10^{\lambda}\sigma^2$.

Assuming the noise being Gaussian distributed, the 68% confidence interval on the exact (intermediate) minimum function value f_{min} is given by $[f_{min} - \alpha\hat{\tau}, f_{min} + \alpha\hat{\tau}]$, where $\alpha = 1$ (95% confidence intervals can be obtained by using $\alpha = 2$). Assuming the intermediate lowest (noisy) function value $\hat{f}_{min}$ is the lower bound, namely, $\hat{f}_{min} = f_{min} - \alpha\hat{\tau}$, then it is easy to see that the upper bound can be expressed in terms of $\hat{f}_{min}$, namely, $f_{min} + \alpha\hat{\tau} = \hat{f}_{min} + 2\alpha\hat{\tau}$. Thus, in sum, assuming that the measurements errors are homogeneous distributed in the input space and the estimated λ is correct, the upper bound is defined by $T_2 = (\hat{f}_{min} + 2\alpha\hat{\tau}) + p \cdot |\hat{f}_{min} + 2\alpha\hat{\tau}|$. Note that this is by no means an unerring formula. While no extra parameters are introduced the estimate of λ may not be accurate and the type of error is often unknown.

Unlike PoI, the gPoI cannot be simply optimized over $\mathbf{x}$ to identify new data samples. Available samples that are already lying in the desired output range $[T_1, T_2]$ have a high probability (see Figure 5.5a), and, hence, straightforward optimization of the criterion might result in duplicate samples. Other (space-filling) strategies have to be devised that makes fully use of the information provided by gPoI. This problem is further explored in the next section.

5.1.3. Search strategies

The techniques explained in the previous sections (e.g., EI, PoI, gPoI, etc.) are all utility functions. Such utility functions are used to identify interesting new points in a sequential design strategy. Various search strategies exist to exploit these utility functions.

For instance, the original EGO algorithm [16] simply optimizes the EI. In particular, the deterministic branch and bound methodology was used

to find the global optimum. To that end, a convex upper bound had to be calculated for the EI. However, if one wants to use other utility functions (or other types of surrogate models) this upper bound must be redefined. In later work more black-box optimization methods were used with similar results, e.g., the DIviding REctangles (DIRECT) [17] or an extensive pattern search. Moreover, multi-modal optimization methods are suggested in literature (e.g., in [19, 23]) to select multiple samples in one iteration, taking full advantage of parallel computing.

Global optimization methods are not suited to directly exploit the new gPoI criterion because there might exist multiple (or even an infinite number of) solutions. Therefore the authors adapt a generic sampling framework for sequential design [7]. The algorithmic flow is depicted in Figure 5.4. The search strategy is configured as followed: First, n candidate samples are drawn from the uniform distribution. Subsequently, these candidates are ranked according to two ($k = 2$) criteria: the gPoI criterion (see Figure 5.5a) and a Minimum Distance (MD) criterion that calculates the Euclidean distance to the closest sample. The MD criterion is defined by,

$$MD(\mathbf{x}) = \frac{(l+1)^{1/d} - 1}{2} \min_{\mathbf{p} \in samples} \sqrt{\sum_{j=1}^{d} (x_j - p_j)^2}, \tag{5.2}$$

where d is the number of input parameters and the factor $\frac{(l+1)^{1/d}-1}{2}$ (upper bound estimate) scales the MD criterion into the same range as the gPoI criterion, namely $[0, 1]$. Assuming the data is scaled to $[-1, 1]$, the estimate on the upper bound is calculated as follows, if the current number of samples is $l + 1$, the optimal maximin configuration of these samples is a uniform grid with $(l + 1)^{(1/d)}$ samples per dimension. Hence, the maximin distance of this layout is a maximum and can be used as an upper bound. Maximizing the MD criterion takes care of the space-filling properties in the input space (see Figure 5.5b). Furthermore, Kriging implies that new input points close to existing samples result in highly correlated output values. Hence, dense clusters of points do not provide much new valuable information and are avoided by the search strategy. The combined sequential sampling criterion is now defined as the weighted average of the two criteria, see Figure 5.5c. Pseudo-code of the QOR sampling algorithm is found in Algorithm 5.1.

This approach is a nice balance between exploration (space-filling) and exploitation (uniform sampling in the input range that satisfies the QORs). If the gPoI criterion is low across the whole input space the MD criterion will dominate and, hence, the input space will be further explored, enhancing the accuracy of the surrogate model. On the other hand, as the number of samples increases, the influence of the MD score will decrease, enabling the exploitation of the gPoI criterion.

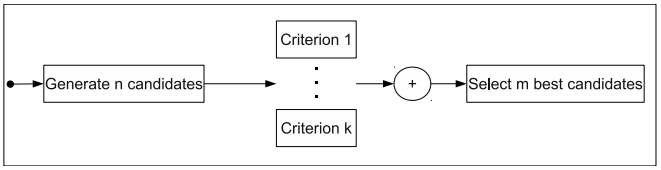
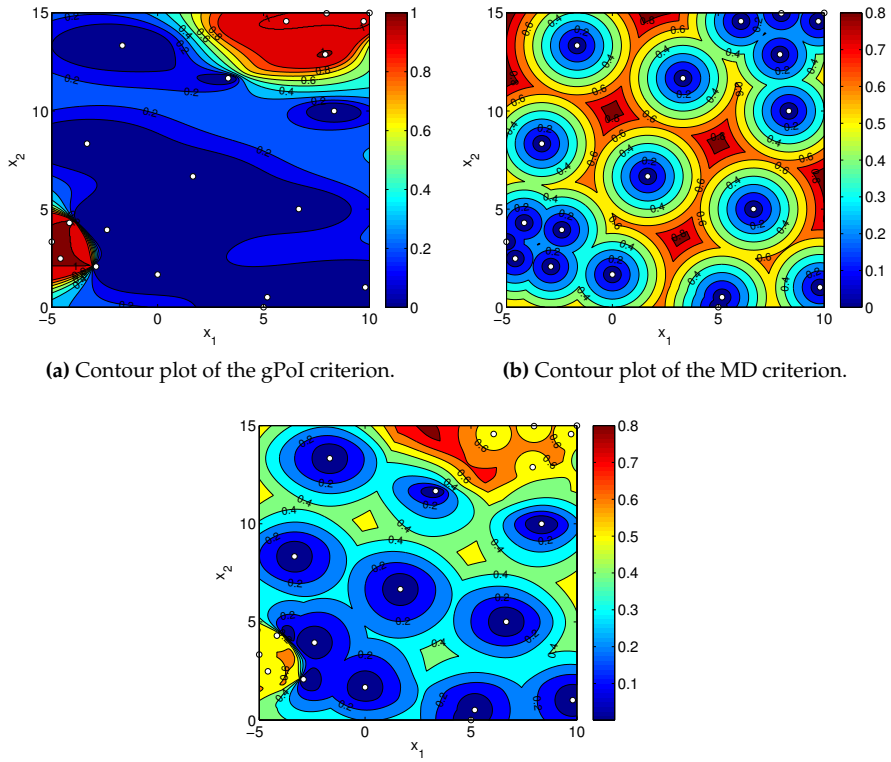


Figure 5.4.: General flow of a sequential design strategy.

Algorithm 5.1 Pseudo-code of the QOR sampling algorithm.

```
1  samples_0 = generateInitialDesign() {e.g., a Latin
       Hypercube Design}
2  values_0 = simulate(samples_0) {call the simulation code}
3
4  T_1 = lowerbound {fraction of the maximal function value
       , e.g., 0.5}
5  T_2 = upperbound {e.g., infinity}
6
7  i = 0
8  while |samples_i| < maxSamples
9    krige_i = fitKriging(samples_i, values_i) {Build surrogate
         model}
10
11   p_test = generateTestPoints(n = 100 × |samples_i|) {Generate n
         test points}
12   score_1 = gPoI(krige_i, p_test, T_1, T_2) {evaluate gPoI}
13   score_2 = MD(p_test) {evaluate the minimum distance}
14   score = w_1 × score_1 + w_2 × score_2 {weighted global score, e.g
         ., w_1 = w_2 = 0.5}
15   p_new = selectBestPoints(p_test, score, m = 10) {select m best
         points from p_test}
16   y_new = simulate(p_new) {call the simulation code}
17
18   samples_{i+1} = samples_i ∪ p_new
19   values_{i+1} = values_i ∪ y_new
20   i = i + 1
21 end
```



(a) Contour plot of the gPoI criterion.

(b) Contour plot of the MD criterion.

(c) Contour plot of the combined criteria.

Figure 5.5.: Example 1: Snapshot of the two sequential sampling criteria (gPoI and MD) and the combined weighted average at 20 samples (white dots) for the 2D Branin function.

5.1.4. Example 1: Determining the QORs of the Branin function

5.1.4.1. Problem setting

The Branin function is a well-known benchmark function for optimization, it has two input variables (x_1, x_2) and its equation is given by,

$$f(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10. \quad (5.3)$$

In this research, the goal is not to identify the unique optimum of the Branin function over the design space, but the goal is to sample the regions corresponding to the 50% highest function values densely in a space-filling

way. So, in this section the QORs correspond with the top 50% of the Branin function.

5.1.4.2. Experimental setup

Version 7.0.2 of the SUMO toolbox is used to determine the QORs of (5.3). An initial set of 10 samples is generated by an optimal maximin Latin Hypercube Design (LHD; [8]). Subsequently, 90 infill points are selected based on the gPoI and MD figures of merit as discussed in Section 5.1.3. To find the QORs, we adapt the bounds $[T_1, T_2]$ of the gPoI criterion in consecute steps, namely, $[T_1, T_2] = [\hat{f}_{max} - 0.5 \cdot |\hat{f}_{max}|, \infty]$, where $\hat{f}_{max}$ is the intermediate maximal function value. Samples are selected in batches of $m = 10$ and the sequential sampling is halted when the number of samples reaches 100. Thus, after the initial set of 10 samples, we have a total of $20, 30, \ldots, 90, 100$ samples. The Kriging surrogate model is configured using the standard Gaussian correlation function and a constant regression function. The hyperparameters, including the λ parameter, are efficiently estimated using SQPLab [4] (`http://www-rocq.inria.fr/~gilbert/modulopt/optimization-routines/sqplab/sqplab.html`), utilizing likelihood derivative information.

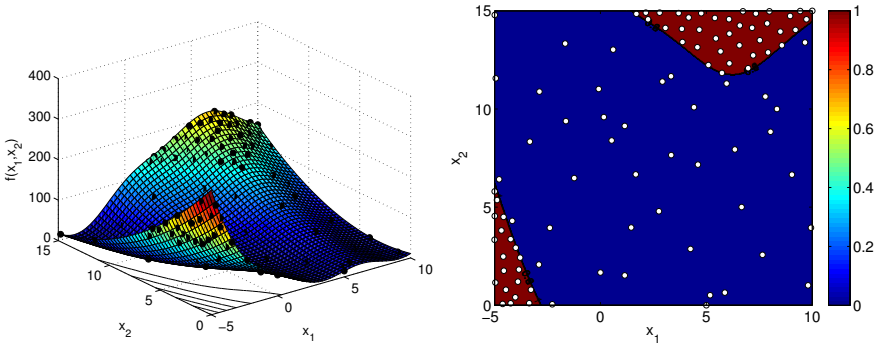
5.1.4.3. Results

An intermediate snapshot of the different sampling criteria (at 20 samples) is shown in Figure 5.5. The gPoI emphasizes the highest regions of the Branin function (i.e., exploiting the function behavior) while the MD criterion takes care of the exploration aspect. The combination of the two criteria is the actual metric used to select new samples in a sequential way. A landscape plot of the final Kriging model and corresponding gPoI contour plot is shown in Figure 5.6. Note that, while the QORs are sampled densely, other regions of the input space have not been neglected completely.

5.1.5. Example 2: Determining the QORs of the Hartman function

5.1.5.1. Problem setting

The six-dimensional Hartman function is another well-known benchmark function for optimization, the Hartman equations are given by,



(a) Landscape plot of the Kriging model.

(b) Contour plot of the gPoI criterion (based on the Kriging model).

Figure 5.6.: Example 1: Final results of the Branin function (100 samples). It is seen that the QORs of interest are densely samples in an uniform way (samples are denoted by the dots).

$$
\begin{aligned}
A &= \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, \\
c &= \begin{pmatrix} 1 & 1.2 & 3 & 3.2 \end{pmatrix}', \\
p &= \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}, \\
f(x_1, \ldots, x_6) &= -\sum_{i=1}^{4} c_i \cdot \exp\left(-\sum_{j=1}^{6} A_{i,j}(x_j - p_{i,j})^2 \right). \qquad (5.4)
\end{aligned}
$$

The goal is to sample the QORs corresponding to the 50% lowest function values densely in a space-filling way.

5.1.5.2. Experimental setup

Version 7.0.2 of the SUMO toolbox is used to determine the QORs of (5.4). An initial set of 51 samples is generated by an optimal maximin Latin Hypercube Design (LHD; [8]). Subsequently, 950 infill points are selected based on the gPoI and MD figures of merit as discussed in Section 5.1.3.

To find the QORs, we adapt the bounds $[T_1, T_2]$ of the gPoI criterion in consecute steps, namely, $[T_1, T_2] = [-\infty, \hat{f}_{min} + 0.5 \cdot |\hat{f}_{min}|]$, where $\hat{f}_{min}$ is the intermediate minimal function value. Samples are selected in batches of $m = 10$ and the sequential sampling is halted when the number of samples exceeds 1000. Hence, after the initial set of 51 samples, we have a total of $61, 71, \ldots, 511, 521, \ldots, 991, 1001$ samples. The Kriging surrogate model is configured using the standard Gaussian correlation function and a constant regression function. The hyperparameters, including the λ parameter, are estimated using SQPLab.

5.1.5.3. Results

The number of samples (in percent) that are inside the QORs versus the number of evaluated samples is given in Figure 5.7. Only 1.9% (= 1 sample) of the initial design of 51 samples satisfies the QORs, using the exact global minimal function value f_{min} in the bound calculation (in contrast to the estimated $\hat{f}_{min}$). As the search progresses the number of samples that satisfies the output range increases rapidly. At 301 samples 53% of the output values (= 160 samples) lie within the desired range. The slight decline from 600 samples onwards is due to the saturation of the QORs with samples, hence, the QOR sampling algorithm starts focusing more on exploring the input domain. A similar trend is observed when using the estimated $\hat{f}_{min}$ to calculate the bounds of the QORs.

Of particular interest is the observation that identifying QORs is less prohibited by the curse of dimensionality than creating a global accurate surrogate model, but more expensive than SBO. While SBO methods only need to evaluate a series of points towards finding the optimum, a global accurate surrogate model needs exponentially more data points to cover the whole input domain, whereas the QOR sampling algorithm only needs enough data points to identify and uniformly cover the QORs.

Figure 5.8 visualizes the final Kriging model. The Figure is arranged as a matrix of contour plots where the contour plots (slices) are of the first two parameters (x_1 and x_2), while the remaining parameters ($x_3, \ldots, x_6$) are set to fixed values. In particular, the parameters x_3 and x_5 are varied in three discrete steps (lower bound, middle, and upper bound of the domain) along the columns of the matrix. Similarly, the parameters x_4 and x_6 are varied along the rows. The samples are denoted by black dots in each contour plot, the larger a dot the closer the sample lies to the actual slice. To avoid clutter, samples lying further than 0.25 from the slice are not shown. It is seen that for $x_4 = 0.5$ and $x_6 = 0$ the Kriging model is more densely sampled, as well at $x_3 = x_4 = x_5 = x_6 = 0.5$. The latter is close (and around) the global optimum, namely, $(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*, x_6^*) = (0.20, 0.15, 0.48, 0.28, 0.31, 0.66)$.

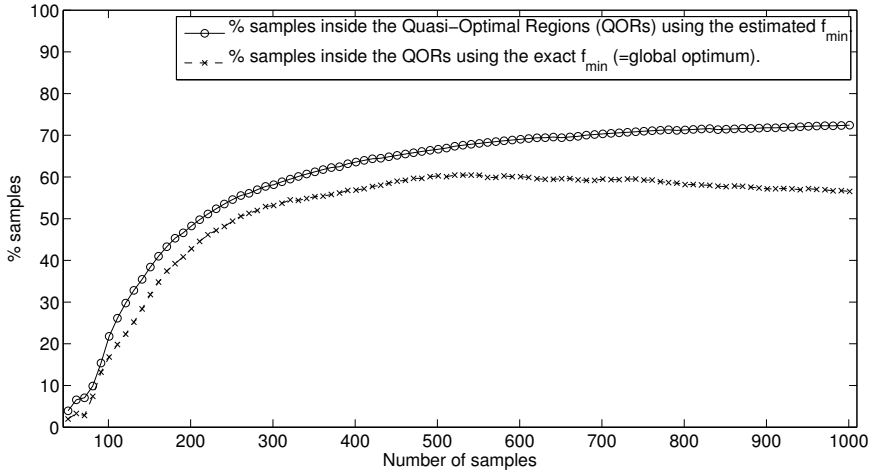


Figure 5.7.: Evolution of the number of samples (in percent) inside the QORs for the Hartman function.

5.1.6. Example 3: Elasticity of the middle ear tympanic membrane

5.1.6.1. Problem setting

In hearing science, finite element modeling is commonly used to study the mechanical behavior of the middle ear, e.g. [12]. In such models, tympanic membrane elasticity parameters have a significant influence on the output [10]. However, good data for the mechanical properties of the tympanic membrane are still lacking [9].

In order to fill this gap, a setup was developed to determine tympanic membrane elasticity in situ by Aernouts et al. [1]. The characterization method consists of four steps: (1) doing a point indentation perpendicular on the membrane surface; (2) measuring the indentation depth, the resulting force and the three-dimensional shape data; (3) simulating the experiment with a finite element model, and (4) adapting the model to fit the measurements using optimization procedures. A detailed description of the application of this method on a rabbit tympanic membrane sample is given in [2].

The tympanic membrane sample (in this case obtained from a rabbit) was placed on a translation and rotation stage, a schematic drawing is shown in Figure 5.9a. Indentations in and out in a direction perpendicular to the surface membrane were carried out using a stepper motor with indentation depths up to 400 micrometer. The resulting force was measured with a load cell and the exact indentation depth was assessed with a Linear Variable Differential Transformer (LVDT).

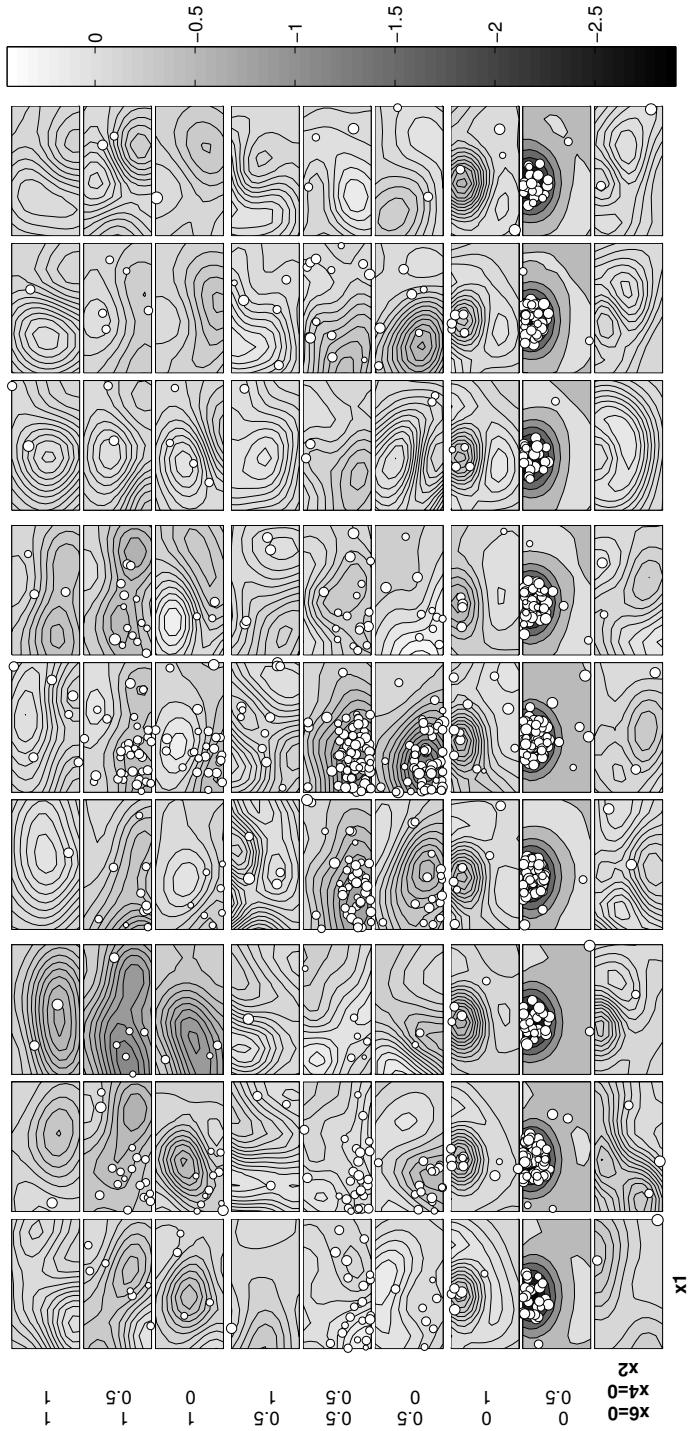


Figure 5.8.: Contour plots arranged in a matrix of the final Kriging model of the Hartman function. The samples are denoted by white dots, the larger a dot the closer the sample lies to the actual slice.

In order to construct a finite element model, an LCD-Moiré profilometer was used to obtain a three-dimensional shape of the membrane before and during indentation. On the basis of these Moiré shape images a highly detailed non-uniform finite element mesh was created. In the needle indentation area and in the manubrium neighborhood, mesh density was increased. This is illustrated in Figure 5.9. The tympanic membranes was modeled as a linear isotropic homogeneous elastic material which is described with two independent elasticity parameters: the Young's modulus E and Poission's ratio ν. The numerical simulations were performed with the finite element code FEBio (`http://mrl.sci.utah.edu/software/febio`), which is specifically designed for bio-mechanical applications.

Determining the value of the linear elasticity parameters is done by minimizing the discrepancy between the model and the experimental measurements. Namely, by calculating,

$$\arg\min_{E,m}(error_{force}), \tag{5.5}$$

where,

$$error_{force} = \frac{1}{N}\sum_{j=1}^{N}(F_{exp}(q_j) - F_{mod}(q_j))^2, \tag{5.6}$$

with N the number of measured points, q_j the indentation depth, $F_{exp}(q_j)$ the experimental force and $F_{mod}(q_j)$ the simulated force.

5.1.6.2. Experimental setup

Version 7.0.2 of the SUMO toolbox is used to determine the QOR of (5.6). An initial set of samples is generated by an optimal maximin Latin Hypercube Design (LHD; [8]) of 7 points together with four corner points, adding up to a total of 11 initial points. Subsequently, infill points are selected based on the gPoI and MD figures of merit. For this problem the upper bound is defined as $T_2 = (\hat{f}_{min} + 2\hat{\tau}) + 0.5 \cdot |\hat{f}_{min} + 2\hat{\tau})|$, namely we are interested in all quasi-optimal solutions that deviate maximally 50% of the best solution found, taking noise into account. Samples are selected and evaluated one by one ($m = 1$) to ensure optimal space-fillingness.

The Kriging surrogate model is configured using the standard Gaussian correlation function and a constant regression function. The hyperparameters, including the λ parameter, are efficiently estimated using SQPLab. The process is halted when the number of samples exceeds 100. The average computation time of FEBio for one simulation is about five to ten minutes.

5.1.6.3. Results

Contour plots of the intermediate Kriging model of $error_{force}$, and the associated gPoI, at various stages in the sampling process are shown in Figure

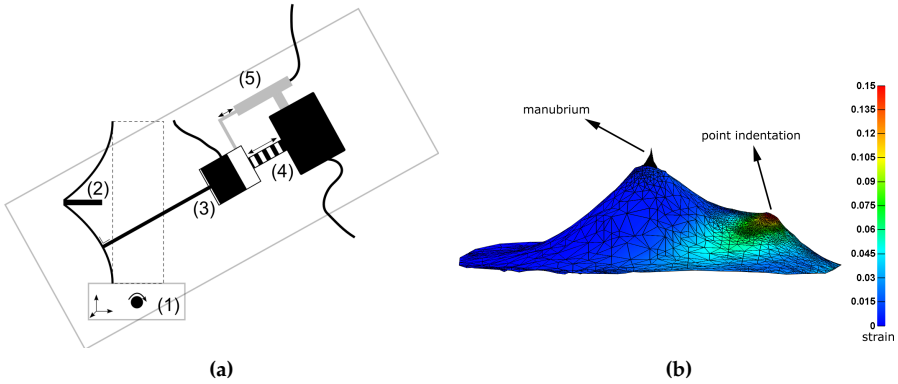


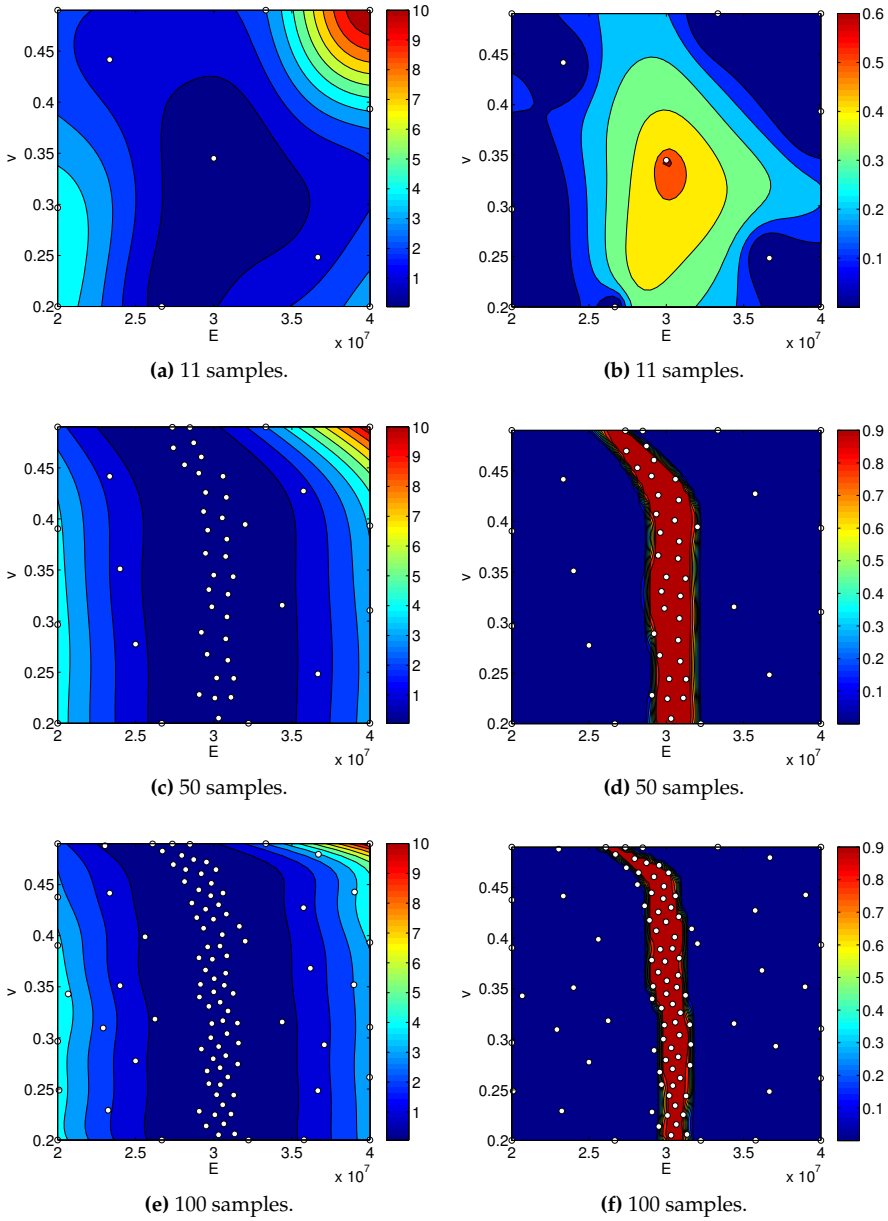
Figure 5.9.: Example 2: Bio-mechanical characterization example. (a) Schematic drawing of the point indentation setup: (1) translation and rotation stage, (2) cross section of tympanic membrane sample, (3) needle connected to a load cell, (4) stepper motor and (5) Linear Variable Differential Transformer. (b) Finite element model of the tympanic membrane with indentation. The number of membrane shell elements equal to 5988. The effective strain in the point indentation area after indentation rises up to approximately 15%.

5.10. The initial Kriging model based on 11 samples has not yet discovered the QOR completely. After a stage of mostly exploration-based sampling, the full QOR is outlined after approximately 50 samples. The focus is now shifted to sampling the identified QOR densely (exploitation) until the stopping criterion of 100 samples is reached.

A contour plot of the final Kriging surrogate model is shown in Figure 5.10e. Obviously, the QOR is quite densely sampled in comparison with other parts of the input domain. Moreover, in Figure 5.10f the contour plot of the gPoI of the final Kriging model shows a clearly defined band in the input domain (= optimal curve) with probability one. The gray zone of uncertainty is reduced to a very small region at the edge of the optimal curve.

5.1.7. Conclusion

This paper introduced a simple but powerful method to solve (inverse) problems consisting of multiple quasi-optimal solutions. The Quasi-Optimal Regions (QORs) are identified with a limited number of expensive function evaluations. The QORs offers the user a trade-off between several solutions, similar to the Pareto front in multiobjective optimization. The QOR sampling method, based on the generalized Probability of Improvement



(a) 11 samples. **(b)** 11 samples.

(c) 50 samples. **(d)** 50 samples.

(e) 100 samples. **(f)** 100 samples.

Figure 5.10.: Example 2: Left column: Contour plots of the intermediate Kriging model of $error_{force}$ at various stages in the sampling process (samples are denoted by dots). The QOR is densely and quasi-uniformly sampled. Right column: Corresponding contour plots of the new gPoI criterion. The QOR is identified properly by the criterion.

(gPoI) criterion, is implemented in the SUMO Matlab toolbox [13], and successfully applied on the Branin and Hartman functions, and used to determine the elasticity of the middle ear tympanic membrane.

Within the QOR sampling algorithm framework several variations are possible. For instance, the gPoI has been successfully applied to identify input regions in the design space that correspond to a certain band in the output space, providing a tool to solve inverse problems directly. Furthermore, the method is relatively dimension-free, i.e., it does not pose any extra restrictions than those already inherent in the Kriging surrogate model.

5.1.8. Bibliography

[1] J. Aernouts, I. Couckuyt, K. Crombecq, and J.J.J. Dirckx. Elastic characterization of membranes with a complex shape using point indentation measurements and inverse modelling. *International Journal of Engineering Science*, 48:599–611, 2010.

[2] J. Aernouts, J. Soons, and J. Dirckx. Quantification of tympanic membrane elasticity parameters from in situ point indentation measurements: Validation and preliminary study. *Hearing Research*, 263:177–182, 2010.

[3] R.R. Barton. Issues in development of simultaneous forward-inverse metamodels. In *Proceedings of the Winter Simulation Conference*, pages 209–217, 2005.

[4] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2006.

[5] I. Couckuyt, F. Declercq, T. Dhaene, and H. Rogier. Surrogate-based infill optimization applied to electromagnetic problems. *Journal of RF and Microwave Computer-Aided Engineering: Advances in design optimization of microwave/rf circuits and systems*, 20(5):492–501, 2010.

[6] I. Couckuyt, D. Gorissen, F. DeTurck, and T. Dhaene. Inverse surrogate modeling: Output performance space sampling. In *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010.

[7] K. Crombecq and T. Dhaene. Generating sequential space-filling designs using genetic algorithms and monte carlo methods. In *Simulated Evolution And Learning (SEAL-2010)*, pages 80–84, 2010.

[8] E.R. Dam, B.G.M. van Husslage, D. den Hertog, and J.B.M. Melissen. Maximin Latin hypercube designs in two dimensions. *Operations Research*, 55(1):158–169, 2007.

[9] W. Decraemer and W. Funnell. *Anatomical and mechanical properties of the tympanic membrane, Chronic Otitis Media. Pathogenesis-Oriented Therapeutic Management*. Kugler Publications, 2008.

[10] N. Elkhouri, H. Liu, and W. Funnell. Low-frequency finite-element modelling of the gerbil middle ear. *Journal of the Association for Research in Otolaryngology*, 7:399–411, 2006.

[11] A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, Chichester, 2008.

[12] R.Z. Gan, Q. Sun, B. Feng, and M.W. Wood. Acoustic-structural coupled finite element analysis for sound transmission in human ear - pressure distributions. *Medical Engineering & Physics*, 28:395–404, 2006.

[13] D. Gorissen, K. Crombecq, I. Couckuyt, P. Demeester, and T. Dhaene. A surrogate modeling and adaptive sampling toolbox for computer based design. *Journal of Machine Learning Research*, 11:2051–2055, 2010.

[14] J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. Technical Report 49–52, Princeton University Bulletin, 1902.

[15] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Global Optimization*, 21:345–383, 2001.

[16] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[17] D.R. Jones, C.D. Perttunen, and B.E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Optimization Theory and Applications*, 79(1):157–181, 1993.

[18] V. Picheny, D. Ginsbourger, O. Roustant, and R.T. Haftka. Adaptive designs of experiments for accurate approximation of a target region. *Mechanical Design*, 132(7):9, 2010.

[19] W. Ponweiser, T. Wagner, and M. Vincze. Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models. In *Congress on Evolutionary Computation*, 2008.

[20] J. Sacks, W. J. Welch, T.J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, 4(4):409–435, 1989.

[21] M.J. Sasena. *Flexibility and Efficiency Enhancements For Constrainted Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

[22] M. Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, 1997.

[23] András Sóbester, Stephen J. Leary, and Andy J. Keane. A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Structural and Multidisciplinary Optimization*, 27:371–383(13), 2004.

[24] J. Staum. Better simulation metamodeling: The why, what, and how of stochastic kriging. In *Proceedings of the Winter Simulation Conference*, 2009.

[25] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

6 Conclusions

Knowledge is soon changed, then lost in the mist, an echo half-heard.
— Gene Wolfe

6.1. Summary

In Chapter 2 the SURogate MOdeling (SUMO) toolbox was presented which offers a flexible framework for surrogate modeling. The SUMO toolbox is a fully featured sequential surrogate modeling framework and was used as a research platform to implement and benchmark several surrogate modeling approaches and algorithms. In the first part of Chapter 2 the use of multiobjective optimization to generate a set of global accurate surrogate models is explored. Subsequently, an object-oriented implementation of the popular Kriging, co-Kriging and blind Kriging surrogate models was presented as the ooDACE toolbox. In contrast to the SUMO toolbox, the ooDACE toolbox offers these types of surrogate models in a small, easy-to-use package that can be integrated in the modeling pipeline of the engineer. The ooDACE toolbox has also been incorporated into the SUMO toolbox as well as all methods developed in this thesis. Both the ooDACE and SUMO toolbox are open source and freely available for academic purposes at the following location `http://sumo.intec.ugent.be` under an open source license (The Affero GNU General Public License[1]), thus encouraging code contribution. While this thesis has focused mainly on the research side of things, the

[1] http://www.fsf.org/licensing/licenses/agpl-3.0.html

development of the publicly available ooDACE and SUMO toolboxes required a substantial investment of time, including maintaining the websites, setting up a test and nightly build infrastructure, writing and updating documentation, fixing bugs, user support, etc. These activities provide reproducibility of the research results and was an excellent means to get in touch with and setup collaborations with the wider research community. The final part of Chapter 2 consists of a thorough benchmark of the blind Kriging surrogate model to show its strength and weaknesses. It is shown that using more complex regression functions enables the application of Kriging to problems with a higher number of dimensions or with gaps in the input space.

The popular Efficient Global Optimization (EGO) algorithm forms the core of this thesis and was applied to novel forward and inverse problems from electromagnetics in Chapter 3. The EGO algorithm was extended to automatically choose the best surrogate model type using the Evolutionary Model Selection (EMS) algorithm. Kriging models, RBF models and SVM models (including ensembles) competed to approximate and optimize a problem from mechanical engineering with interesting results. However, the computational cost associated with the EMS algorithm is quite high. Hence, it may be more useful to construct several surrogate models in parallel and afterwards select the best surrogate model (instead of using evolution).

The EGO algorithm is quite often used for single-objective optimization. While attempts are made to adapt EGO and its statistical criteria for multiobjective optimization, the calculation of multiobjective criteria is much more difficult for many objectives. In Chapter 4 an algorithm was presented to make the calculation of the criteria for many objectives feasible up to 8 objectives. Different types of statistical criteria were benchmarked for multiobjective optimization on a test suite of functions. The algorithm was found to consistently outperform several multiobjective evolutionary algorithms. While the presented method for calculating the criteria is very fast and stable, more work is needed on improving the quality of the surrogate model (which is crucial) and on efficiently optimizing the difficult multi-modal landscape of the statistical criteria.

Chapter 5 introduced a novel algorithm to directly solve inverse problems by sampling the regions in the input space that correspond to the desired output value(s). Similarly to Pareto optimization techniques, this approach identifies multiple possible solutions allowing the decision maker to make more informed decisions for the final selected solution.

Beside the minor limitations and possible improvements already mentioned, two larger future work directions can be distinguished which are further discussed in the remainder of this Chapter.

6.2. Future work

6.2.1. Medium-scale and large-scale problems

The presented methods in this thesis are mature and can be safely used by engineers for many problems. The limitations of the developed algorithms lies mostly in the quality of the surrogate model. The main challenge in engineering is the "*curse of dimensionality*" and perhaps this is even more true for methods based on surrogate models [11]. Popular surrogate modeling methods are only capable of handling relatively small-scale problems (<20 dimensions). By restricting the use of surrogate models to local parts of the design space this can be slightly alleviated to medium-scale (20-100 dimensions) problems, e.g., using a trust region-like mechanism [1]. However, this negates the benefit of surrogate models providing a global overview of the problem (e.g., to identify global optima). Of course, medium-scale (20-100 dimensions) and large-scale (>100 dimensions) problems are of great interest in industrial applications.

Hence, the most important enhancements to Kriging-based methods, as those presented in this thesis, involve their ability to solve medium-scale and large-scale problems. This includes, but is not limited to high-dimensionality, many (expensive) constraints, large input spaces, large datasets, and complex non-linear behavior of the computational expensive simulator. Current algorithms are restricted to low-dimensional spaces (<20 dimensions), a small training set (<1000 samples) and are only suited for relatively smooth functions. Of course, this is a rather large problem and, thus, should be addressed on several fronts.

A first step would be to scale up to medium-scale problems. To that end, the computational complexity of the Kriging model itself needs to be reduced. This can be achieved by improving the underlying mathematical framework using, e.g., use (low-rank) approximations of the covariance matrix (better scaling with the number of samples), likelihood approximations, localized regression, non-stationary covariance functions [13] (more complex behaving functions), efficient update formulas [4], and efficient optimization routines for the maximum likelihood estimation [14] (increases the dimensions). While some initial research has been done on improving the Kriging model, they often have restrictions, e.g., Monte Carlo approximations or a number of free (or magic) parameters, and many are only tested separately on relatively simple mathematical examples. Moreover, the performance benefits of these improvements to actual optimization, sensitivity analysis, etc. problems remains largely unknown as they have only been tested on their approximation capability and not necessarily the performance of the broader algorithm where the Kriging models are used. Surrogate models in optimization, global reliability analysis, sensitivity analysis, etc. do not need to have the best accuracy, but just enough so to guide the search towards its goal.

In a second step, beside improving the Kriging model, a further increase in dimensionality can be gained by using feature (or variable) selection methods [5]. In high dimensional spaces, efficient algorithms for feature selection are of paramount importance to reduce the computational cost and allow a focused investigation of the relevant system design variables. Usually, not all of these variables are relevant for the problem at hand, and several sets of variables might be correlated. Therefore, feature selection methods are applied to simplify the data. Irrelevant variables may be either left out completely (dimension reduction) or aggregated into variable subsets (known as a *features*). This results in an updated dataset, expressed in terms of these features, which is easier to analyze. The machine learning community is particularly active in the domain of feature selection methods as they often have to deal with enormous amounts of high dimensional data. Most research is focused on classification techniques, however, a lot of these techniques are also applicable to regression problems. It should be noted that the problem of selection the right features in regression is more commonly known as variable subset selection or subset selection in regression [7]. Feature selection methods can be applied as a pre-processing step or integrated into the Kriging model and the Kriging-based methods. The latter approach has the benefit that during, e.g., optimization (ir)relevant variables can be included or left out automatically.

The third and final step addresses large-scale problems. By decomposing the initial problem into sub-problems, multiple lower-dimensional Kriging submodels can be built using the improvements obtained from the first step. These lower-dimensional submodels are then used as building blocks within a high dimensional model representation (HDMR) [10, 12]. The HDMR expands a function with n-dimensional input to summands and/or multiplications of different sub-functions of less than n dimensions.

6.2.2. Broaden class of problems

Kriging provides a mathematical framework to efficiently cope with a diverse set of simulation problems. Different kinds of computational expensive simulators exist, each requiring a distinct set of methods. Kriging models are universal, in a sense that they can be extended to take full advantage of certain properties of the simulator, increasing its efficiency in a mathematical sound way. In literature there have already been many extensions to Kriging to handle different simulation properties, including multi-fidelity simulation [6], multi-output approximation [3, 2] (taking full advantage of the correlation between outputs), partially converging simulations [9] (as data is expensive everything counts), continuous and discrete outputs, noise on the output, utilizing gradient information in the prediction, expensive constraint handling [8], etc. Others, such as complex outputs, have not been investigated yet. Many of these extensions are promising but have not yet been thoroughly tested and often have only been

shown to increase the accuracy of Kriging regression for approximations problems. Their usefulness to expedite specific time-consuming tasks such as optimization and sensitivity analysis have not always been proved. In their current state they each have their drawbacks such as increased computational complexity or making too many assumptions. Most importantly, each extension to the Kriging model has been proposed and tested on its own. More complex problems that need a combination of these extensions, e.g., multi-fidelity and mixed-integer optimization, have not been tackled yet. Further research is warranted to make each of these extensions viable and combine them into one Kriging model - a straightforward combination is not always possible due to possible conflicts in the underlying mathematical model - so that a broad class of very complex problems can be solved.

More emphasis needs to be given to the versatility of the designed Kriging algorithms. It is of paramount importance to create an algorithm that can handle a variety of problems with different properties, even though they do not necessarily yield the best performance on each problem type. There is an increasing need for algorithms that guarantee convergence to a *good* solution for a wide range of problems rather than a specific algorithm that obtains the optimal solution, though only on a very small set of problems. Kriging-based methods should hold up well when dealing with mixed-integer problems, different types of constraints (soft/hard, (in)equal), noise on the output, multiple correlated outputs, complex outputs, etc. In each case the algorithm should be able to take full advantage of the properties of the simulator to converge to a good optimal solution.

6.3. Bibliography

[1] N. Alexandrov, J. E. Dennis, R. M. Lewis, and V. Torczon. A trust region framework for managing the use of approximation models in optimization. *Structural Optimization*, 15:16–23, October 1998.

[2] M. A. Alvarez, D. Luengo, M. K. Titsias, and N. D. Lawrence. Efficient multioutput gaussian process through variational inducing kernels. In *Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

[3] P. Boyle and M. Frean. *Advances in Neural Information Processing Systems 17*, chapter Dependent Gaussian processes, pages 217–224. The MIT Press, 2005.

[4] C. Chevalier and D. Ginsbourger. Corrected kriging update formulae for batch-sequential data assimilation. submitted.

[5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Machine Learning*, 3:1157–1182, 2003.

[6] M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87:1–13, 2000.

[7] A.J. Miller. *Subset Selection in Regression*. London: Chapman & Hall, 1990.

[8] J. Parr, A.J. Keane, A.I.J. Forrester, and C.M.E. Holden. Infill sampling criteria for surrogate-based optimization with constraint handling. *Engineering Optimization*, 44(10):1147–1166, 2012.

[9] V. Picheny and D. Ginsbourger. Space-time gaussian processes for the approximation of partially converged simulations. *SIAM/ASA Journal on Uncertainty Quantification.*, submitted.

[10] S. Shan and G. Wang. Development of adaptive rbf-hdmr model for approximating high dimensional problems. In *Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2009, San Diego, USA, DETC2009-86531*, 2009.

[11] S. Shan and G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41:219–241, 2010.

[12] S. Shan and G. Wang. Turning black-box functions into white functions. *Journal of Mechanical Design*, 133(3):1–10, March 2011.

[13] D. J.J. Toal and A. J. Keane. Non-stationary kriging for design optimization. *Engineering Optimization*, 44(6):741–765, 2012.

[14] D.J.J. Toal, A. I.J. Forrester, N.W. Bresslof, A.J. Keane, and C. Holden. An adjoint for likelihood maximization. *Royal Society*, pre-print, 2009.

A

SUMO Toolbox

A reader is not supposed to be aware that someone's written the story. He's supposed to be completely immersed, submerged in the environment.
— Jack Vance

A.1. A Surrogate Modeling and Adaptive Sampling Toolbox for Computer Based Design

D. Gorissen, K. Crombecq, I. Couckuyt, P. Demeester, T. Dhaene

Published in Journal of Machine Learning Research,

vol. 11, pp. 2051-2055, 2010.

Abstract

An exceedingly large number of scientific and engineering fields are confronted with the need for computer simulations to study complex, real world phenomena or solve challenging design problems. However, due to the computational cost of these high fidelity simulations, the use of neural networks, kernel methods, and other surrogate modeling techniques have

become indispensable. Surrogate models are compact and cheap to evaluate, and have proven very useful for tasks such as optimization, design space exploration, prototyping, and sensitivity analysis. Consequently, in many fields there is great interest in tools and techniques that facilitate the construction of such regression models, while minimizing the computational cost and maximizing model accuracy. This paper presents a mature, flexible, and adaptive machine learning toolkit for regression modeling and active learning to tackle these issues. The toolkit brings together algorithms for data fitting, model selection, sample selection (active learning), hyperparameter optimization, and distributed computing in order to empower a domain expert to efficiently generate an accurate model for the problem or data at hand.

A.1.1. Background and Motivation

In many science and engineering problems researchers make heavy use of computer simulation codes in order to replace expensive physical experiments and improve the quality and performance of engineered products and devices. Such simulation activities are collectively referred to as computational science/engineering. Unfortunately, while allowing scientists more flexibility to study phenomena under controlled conditions, computer simulations require a substantial investment of computation time. One simulation may take many minutes, hours, days or even weeks, quickly rendering parameter studies impractical [2, 9].

Of the different ways to deal with this problem, this paper is concerned with the construction of simpler approximation models to predict the system performance and develop a relationship between the system inputs and outputs. When properly constructed, these approximation models mimic the behavior of the simulation accurately while being computationally cheap(er) to evaluate. Different approximation methods exist, each with their relative merits. This work concentrates on the use of data-driven, global approximations using compact surrogate models (also known as metamodels, replacement models, or response surface models). Examples include: rational functions, Kriging models, Artificial Neural Networks (ANN), splines, and Support Vector Machines (SVM). Once such a global approximation is available it is of great use for gaining insight into the behavior of the underlying system. The surrogate may be easily queried, optimized, visualized, and seamlessly integrated into CAD/CAE software packages.

The challenge is thus how to generate an approximation model that is as accurate as possible over the *complete* domain of interest while minimizing the simulation cost. Solving this challenge involves multiple sub-problems that must be addressed: how to interface with the simulation code, how to run simulations (locally, or on a cluster or cloud), which model type to approximate the data with and how to set the model complexity (e.g.,

topology of a neural network), how to estimate the model quality and ensure the domain expert trusts the model, how to decide which simulations to run (data collection), etc. The data collection aspect is worth emphasizing. Since data is computationally expensive to obtain and the optimal data distribution is not known up front, data points should be selected iteratively, there where the information gain will be the greatest. A sampling function is needed that minimizes the number of sample points selected in each iteration, yet maximizes the information gain of each iteration step. This process is called adaptive sampling but is also known as active learning, or sequential design.

There is a complex dependency web between these different options and dealing with these dependencies is non-trivial, particularly for a domain expert for whom the surrogate model is just an intermediate step towards solving a larger, more important problem. Few domain experts will be experts in the intricacies of efficient sampling and modeling strategies. Their primary concern is obtaining an accurate replacement metamodel for their problem as fast as possible and with minimal overhead [7]. As a result these choices are often made in a pragmatic, sometimes even ad-hoc, manner.

This paper discusses an advanced, and integrated software framework that provides a flexible and rigorous means to tackle such problems. This work lies at the intersection of Machine Learning/AI, Modeling and Simulation, and Distributed Computing. The methods developed are applicable to any domain where a cheap, accurate, approximation is needed to replace some expensive reference model. Our experience has been that the availability of such a framework can facilitate the transfer of knowledge from surrogate modeling researchers and lower the barrier of entry for domain experts.

A.1.2. SUMO Toolbox

The platform in question is the Matlab SUrrogate MOdeling (SUMO) Toolbox, illustrated in Figure A.1. Given a simulation engine (Fluent, Cadence, Abaqus, HFSS, etc.) or other data source (data set, Matlab script, Java class, etc.), the toolbox drives the data source to produce a surrogate model within the time and accuracy constraints set by the user.

The SUMO Toolbox adopts a microkernel design philosophy with many different plugins available for each of the different sub-problems[1]: model types (rational functions, Kriging, splines, SVM, ANN, etc.), hyperparameter optimization algorithms (Particle Swarm Optimization, Efficient Global Optimization, simulated annealing, Genetic Algorithm, etc.), model selection algorithms (cross validation, AIC, Leave-out set, etc.), sample selection (random, error based, density based, hybrid, etc.), Design of Experiments

[1] The full list of plugins and features can be found at `http://www.sumowiki.intec.ugent.be`

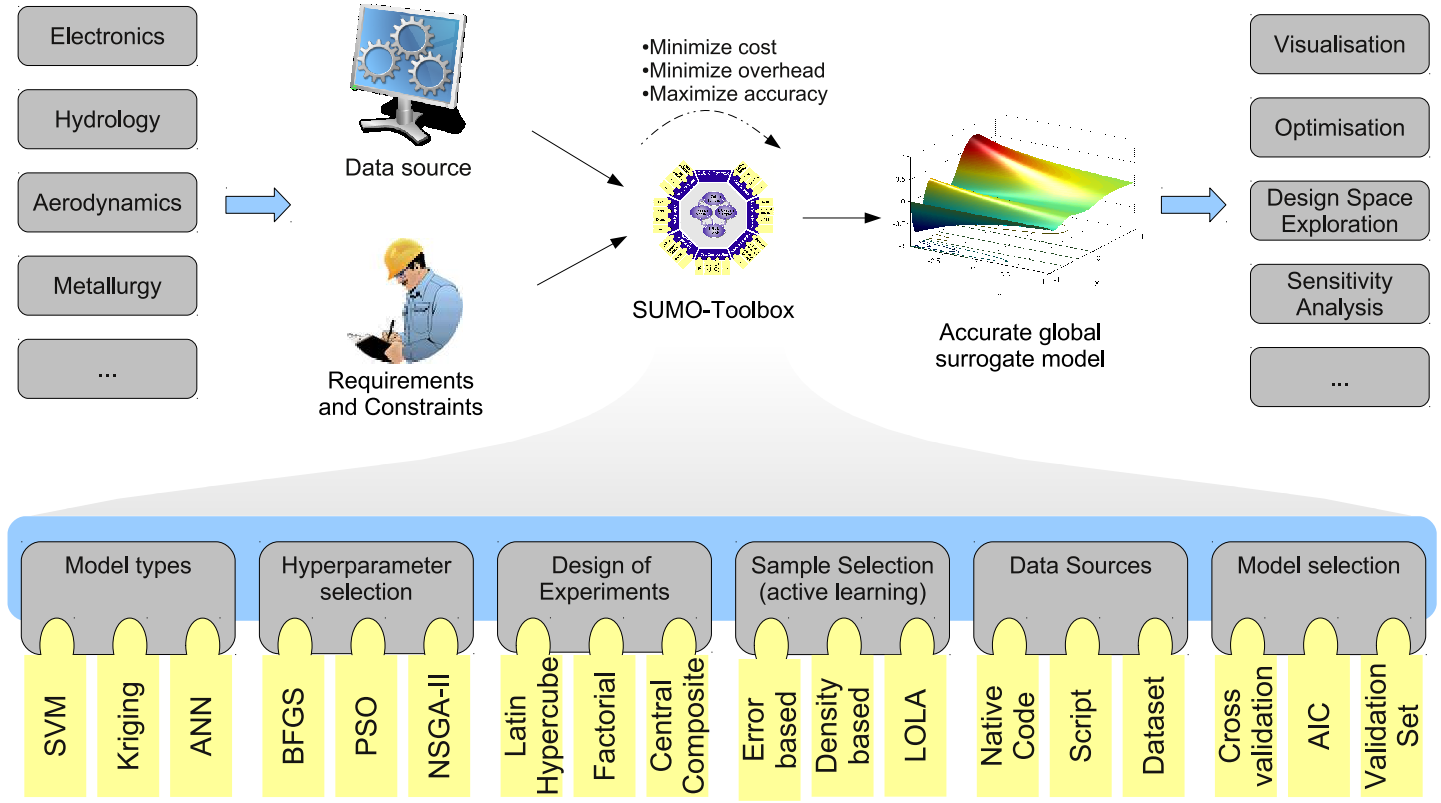


Figure A.1.: The SUMO Toolbox is a flexible framework for accurate global surrogate modeling and adaptive sampling (active learning). It features a rich set of plugins, is applicable to a wide range of domains, and can be applied in an autonomous, black-box fashion, or under full manual control. Written in Matlab and Java it is fully cross platform and comes with a large (60+) number of example problems.

(Latin hypercube, Box-Bhenken, etc.), and sample evaluation methods (local, on a cluster or grid). The behavior of each software component is configurable through a central XML file and components can easily be added, removed or replaced by custom implementations. In addition the toolbox provides 'meta' plugins. For example to automatically select the best model type for a given problem [7] or to use multiple model selection or sample selection criteria in concert [3].

Furthermore, there is built-in support for high performance computing. On the modeling side, the model generation process can take full advantage of multi-core CPUs and even of a complete cluster or grid. This can result in significant speedups for model types where the fitting process can be expensive (e.g., neural networks). Likewise, sample evaluation (simulation) can occur locally (with the option to take advantage of multi-core architectures) or on a separate compute cluster or grid (possibly accessed through a remote head-node). All interfacing with the grid middleware (submission, job monitoring, rescheduling of failed/lost simulation points, etc.) is handled transparently and automatically (see [6] for more details). Also, the sample evaluation component runs in parallel with the other components (non-blocking) and not sequentially. This allows for an optimal use of computational resources.

In addition the SUMO Toolbox contains extensive logging and profiling capabilities so that the modeling process can easily be tracked and the modeling decisions understood. Once a final model has been generated, a GUI tool is available to visually explore the model (including derivatives and prediction uncertainty), assess its quality, and export it for use in other software tools.

A.1.3. Applications

The SUMO Toolbox has already been applied successfully to a very wide range of applications, including RF circuit block modeling [5], hydrological modeling [1], Electronic Packaging [11], aerodynamic modeling [4], process engineering [10], and automotive data modeling [3]. Besides global modeling capabilities, the SUMO Toolbox also includes a powerful optimization framework based on the Efficient Global Optimization framework developed by Jones [8]. As of version 6.1, the toolbox also contains an example of how the framework can also be applied to solve classification problems.

In sum, the goal of the toolbox is to fill the void in machine learning software when it comes to the challenging, costly, real-valued, problems faced in computational engineering. The toolbox is in use successfully at various institutions and we are continuously refining and extending the set of available plugins as the number of applications increase. Usage instructions, design documentation, and stable releases for all major platforms can be found at `http://www.sumo.intec.ugent.be`.

A.1.4. Bibliography

[1] I. Couckuyt, D. Gorissen, H. Rouhani, E. Laermans, and T. Dhaene. Evolutionary regression modeling with active learning: An application to rainfall runoff modeling. In *International Conference on Adaptive and Natural Computing Algorithms*, volume LNCS 5495, pages 548–558, Sep. 2009.

[2] A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, Chichester, 2008.

[3] D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene. Multiobjective global surrogate modeling,dealing with the 5-percent problem. *Engineering with Computers*, 26(1):81–89, Jan. 2010.

[4] D. Gorissen, K. Crombecq, I. Couckuyt, and T. Dhaene. *Foundations of Computational Intelligence, Volume 1: Learning and Approximation: Theoretical Foundations and Applications*, volume 201, chapter Automatic Approximation of Expensive Functions with Active Learning, pages 35–62. Springer Verlag, Series Studies in Computational Intelligence, 2009.

[5] D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene. Sequential modeling of a low noise amplifier with neural networks and active learning. *Neural Computing and Applications*, 18(5):485–494, Jun. 2009.

[6] D. Gorissen, T. Dhaene, P. Demeester, and J. Broeckhove. *Handbook of Research on Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Applications*, chapter Grid enabled surrogate modeling, pages 249–258. IGI Global, May 2009.

[7] D. Gorissen, T. Dhaene, and F. DeTurck. Evolutionary model type selection for global surrogate modeling. *Journal of Machine Learning Research*, 10:2039–2078, 2009.

[8] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[9] T. W. Simpson, V. Toropov, V. Balabanov, and F. A. C. Viana. Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come or not. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008 MAO, Victoria, Canada*, 2008.

[10] D.W. Stephens, D. Gorissen, and T. Dhaene. Surrogate based sensitivity analysis of process equipment. In *Proc. of 7th International Conference on CFD in the Minerals and Process Industries, CSIRO, Melbourne, Australia*, Dec. 2009.

[11] T. Zhu and P. D. Franzon. Application of surrogate modeling to generate compact and PVT-sensitive IBIS models. In *Proceedings of the 18th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct. 2009.

B
ooDACE Toolbox

Indeed Bugg. is it because, do you think, at the human core, we are naught but liars and cheats?"
"Probably."
"With no hope of ever overcoming our instinctive nastiness?"
"Hard to say. How have we done so far?"
— Steven Erikson

B.1. ooDACE Toolbox: A Flexible Object-Oriented Kriging Implementation

I. Couckuyt, T. Dhaene

Accepted for publication in the Journal of Machine Learning Research.

Abstract

When analyzing data from computationally expensive simulation codes, surrogate modeling methods are firmly established as facilitators for design space exploration, sensitivity analysis, visualization and optimization. Kriging is a popular surrogate modeling technique used for the Design and Analysis of Computer Experiments (DACE). Hence, the past decade Kriging has been the subject of extensive research and many extensions have been

proposed, e.g., co-Kriging, stochastic Kriging, blind Kriging, etc. However, few Kriging implementations are publicly available and tailored towards scientists and engineers. Furthermore, no Kriging toolbox exists that unifies several Kriging flavors. This paper addresses this need by presenting an efficient object-oriented Kriging implementation and several Kriging extensions, providing a flexible and easily extendable framework to test and implement new Kriging flavors while reusing as much code as possible.

B.1.1. Introduction

This paper is concerned with efficiently solving complex, computational expensive problems using surrogate modeling techniques [13, 5]. Surrogate models, also known as metamodels, are cheap approximation models for computational expensive (black-box) simulations. Surrogate modeling techniques are well-suited to handle, for example, expensive finite element (FE) simulations and computational fluid dynamic (CFD) simulations.

Kriging is a popular surrogate model type to approximate deterministic noise-free data. First conceived by Danie Krige in geostatistics [7] and later introduced for the Design and Analysis of Computer Experiments (DACE) by Sacks et al. [11], these Gaussian Process [10] based surrogate models are compact and cheap to evaluate, and have proven to be very useful for tasks such as optimization [6], design space exploration, visualization, prototyping, and sensitivity analysis [13, 4]. Note that Kriging surrogate models are primarily known as Gaussian Processes in the machine learning community. Except for the utilized terminology there is no difference between the terms and associated methodologies. Hence, for clarity the term Kriging is used in this work.

While Kriging is a popular surrogate model type, not many publicly available, easy-to-use Kriging implementations exist. Many Kriging implementations are outdated, provided as demo code with a book and/or limited to one specific type of Kriging. Perhaps the most well-known Kriging toolbox is the DACE toolbox[1] of Lophaven et al. [8], but, unfortunately, the toolbox has not been updated for some time and only the standard Kriging model is provided. Other freely available Kriging codes include: stochastic Kriging [12] [2], DiceKriging [3], Gaussian Processes for Machine Learning [9] (GPML)[4], demo code provided with [3][5] and the Matlab Kriging toolbox[6].

This paper addresses this need by presenting an efficient object-oriented Kriging implementation and several Kriging extensions, providing a flexible

[1] http://www2.imm.dtu.dk/~hbn/dace/
[2] http://stochasticKriging.net/
[3] http://cran.r-project.org/web/packages/DiceKriging/index.html
[4] http://mloss.org/software/view/263/
[5] http://www.wiley.com//legacy/wileychi/forrester/
[6] http://globec.whoi.edu/software/Kriging/V3/english.html

and easily extendable framework to test and implement new Kriging flavors while reusing as much code as possible.

B.1.2. ooDACE Toolbox

The ooDACE toolbox is an object-oriented Matlab toolbox implementing a variety of Kriging flavors and extensions. The most important features and Kriging flavors include,

- Simple Kriging, ordinary Kriging, universal Kriging, stochastic Kriging (regression Kriging), blind- and co-Kriging.
- Re-interpolation of the prediction variance (for regression Kriging).
- Derivatives of the prediction and prediction variance.
- Flexible hyperparameter optimization.
- Useful utilities include: cross-validation, integrated mean squared error, empirical variogram plot, debug plot of the likelihood surface, robustness-criterion value, etc.
- Proper object-oriented design (compatible interface with the DACE toolbox[1] is available).

Documentation of the ooDACE toolbox is provided in the form of a getting started guide (for users), a wiki[7] and doxygen documentation[8] (for developers and more advanced users). In addition, the code is well-documented, providing references to research papers where appropriate. A quick-start demo script is provided with five surrogate modeling use cases, as well as script to run a suite of regression tests.

A simplified UML class diagram, showing only the most important public operations, of the toolbox is shown in Figure B.1. The toolbox is designed with efficiency and flexibility in mind. The process of constructing (and predicting) a Kriging model is decomposed in several smaller, logical steps, e.g., constructing the correlation matrix, constructing the regression matrix, updating the model, optimizing the parameters, etc. These steps are linked together by higher-level steps, e.g., fitting the Kriging model and making predictions. The basic steps needed for Kriging are implemented as (protected) operations in the BasicGaussianProcess superclass. Implementing a new Kriging type, or extending an existing one, is now done by subclassing the Kriging class of your choice and inheriting the (protected) methods that need to be reimplemented. Similarly, to implement a new hyperparameter optimization strategy it suffices to create a new class inherited from the Optimizer class.

[7] `http://sumowiki.intec.ugent.be/index.php/ooDACE:ooDACE_toolbox`
[8] `http://sumo.intec.ugent.be/buildbot/doc/`

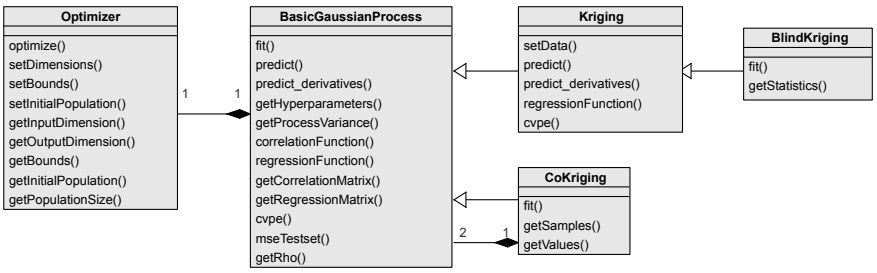


Figure B.1.: Class diagram of the ooDACE toolbox.

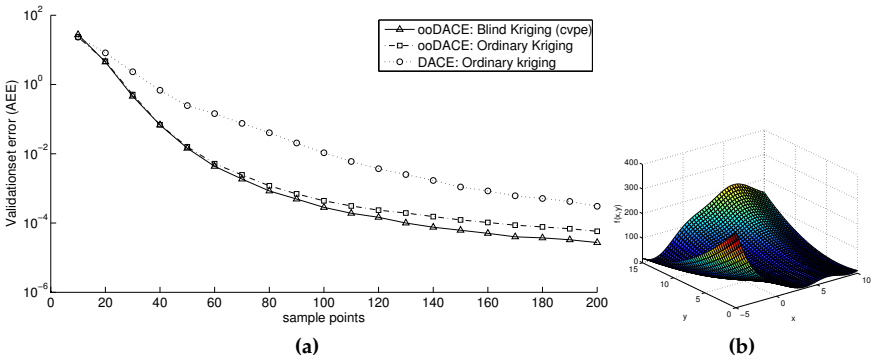


Figure B.2.: (a) Evolution of the average AEE versus the number of samples (Branin function). (b) Landscape plot of the Branin function.

To assess the performance of the ooDACE toolbox a comparison between the ooDACE toolbox and the DACE toolbox[1] is performed using the 2D Branin function. To that end, 20 datasets of increasing size are constructed, each drawn from an uniform random distribution. The number of observations ranges from 10 to 200 samples with steps of 10 samples. For each dataset, a DACE toolbox[1] model, a ooDACE ordinary Kriging and a ooDACE blind Kriging model have been constructed and the accuracy is measured on a dense test set using the Average Euclidean Error (AEE). Moreover, each test is repeated 1000 times to remove any random factor, hence the average accuracy of all repetitions is used. Results are shown in Figure B.2a. Clearly, the ordinary Kriging model of the ooDACE toolbox consistently outperforms the DACE toolbox for any given sample size, mostly due to a better hyperparameter optimization, while the blind Kriging model is able improve the accuracy even more.

B.1.3. Applications

The ooDACE Toolbox has already been applied successfully to a wide range of problems, e.g., optimization of a textile antenna [2], identification of the elasticity of the middle-ear drum [1], etc.

In sum, the ooDACE toolbox aims to provide a modern, up to date Kriging framework catered to scientists and engineers. Usage instructions, design documentation, and stable releases can be found at `http://sumo.intec.ugent.be/?q=ooDACE`.

B.1.4. Bibliography

[1] J. Aernouts, I. Couckuyt, K. Crombecq, and J.J.J. Dirckx. Elastic characterization of membranes with a complex shape using point indentation measurements and inverse modelling. *International Journal of Engineering Science*, 48:599–611, 2010.

[2] I. Couckuyt, F. Declercq, T. Dhaene, and H. Rogier. Surrogate-based infill optimization applied to electromagnetic problems. *Journal of RF and Microwave Computer-Aided Engineering: Advances in design optimization of microwave/rf circuits and systems*, 20(5):492–501, 2010.

[3] A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, Chichester, 2008.

[4] A. I.J. Forrester and A. J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45:50–79, 2009.

[5] D. Gorissen, K. Crombecq, I. Couckuyt, P. Demeester, and T. Dhaene. A surrogate modeling and adaptive sampling toolbox for computer based design. *Journal of Machine Learning Research*, 11:2051–2055, 2010.

[6] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[7] D.G. Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the chemical, metallurgical and mining society of South Africa*, 52:119–139, 1951.

[8] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.

[9] C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

[10] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[11] J. Sacks, W. J. Welch, T.J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, 4(4):409–435, 1989.

[12] J. Staum. Better simulation metamodeling: The why, what, and how of stochastic kriging. In *Proceedings of the Winter Simulation Conference*, 2009.

[13] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

B.2. Practical guide

B.2.1. Download

See the download page at `http://sumo.intec.ugent.be/?q=ooDACE`.

Requirements The ooDACE toolbox takes advantage of the new Object Oriented system (*classdef*) available from Matlab 2008b (v7.7) onwards. By default, the *oodacefit* and *demo* scripts of the ooDACE toolbox use the *fmincon* optimization routine from the Matlab Optimization toolbox. Support for other optimization strategies can easily be used if a wrapper class is coded for it, see Figure B.4. To that end, full support for the third-party SQPLab optimization package [1] (`http://www-rocq.inria.fr/~gilbert/modulopt/optimization-routines/sqplab/sqplab.html`) is included as well as support for the genetic algorithm of the Matlab Global Optimization toolbox.

B.2.2. Getting started

> Before the ooDACE toolbox can be used you have to include it in Matlab's search path. You can do this manually by running startup, or, if Matlab is started in the root toolbox directory, then startup will be run automatically.

```
startup
```

Now the toolbox is ready to be used. The ooDACE toolbox is designed in an Object Oriented (OO) fashion. It is strongly recommended to exploit the OO design directly, i.e., use the Kriging and Optimizer matlab classes, see Figures B.3 and B.4. Most functionality is implemented in the base class *BasicGaussianProcess*, the derived *Kriging* class differs only in the fact that it automatically normalizes your dataset before fitting. For more information on the classes and their methods please refer to the doxygen documentation and the source files.

Lets define n as the number of observations and d as the number of input parameters. Then the $n \times d$ input sample matrix is denoted by **samples** (each row is one observation) and the corresponding output values are stored in the $n \times 1$ matrix **values**. The ooDACE toolbox provides a script, *oodacefit.m*, that just takes your dataset (a **samples** and **values** matrix) and, optionally, an options structure and returns a fitted Kriging object, all other parameters are set to some sensible defaults (the options structure is merged with the defaults). The internal call hierarchy of fitting a Kriging object, and predicting some points, is shown by a sequence diagram in Figure B.5, any part of the fitting (or prediction) process can be easily modified by inheriting from the appropriate class and overriding the desired methods.

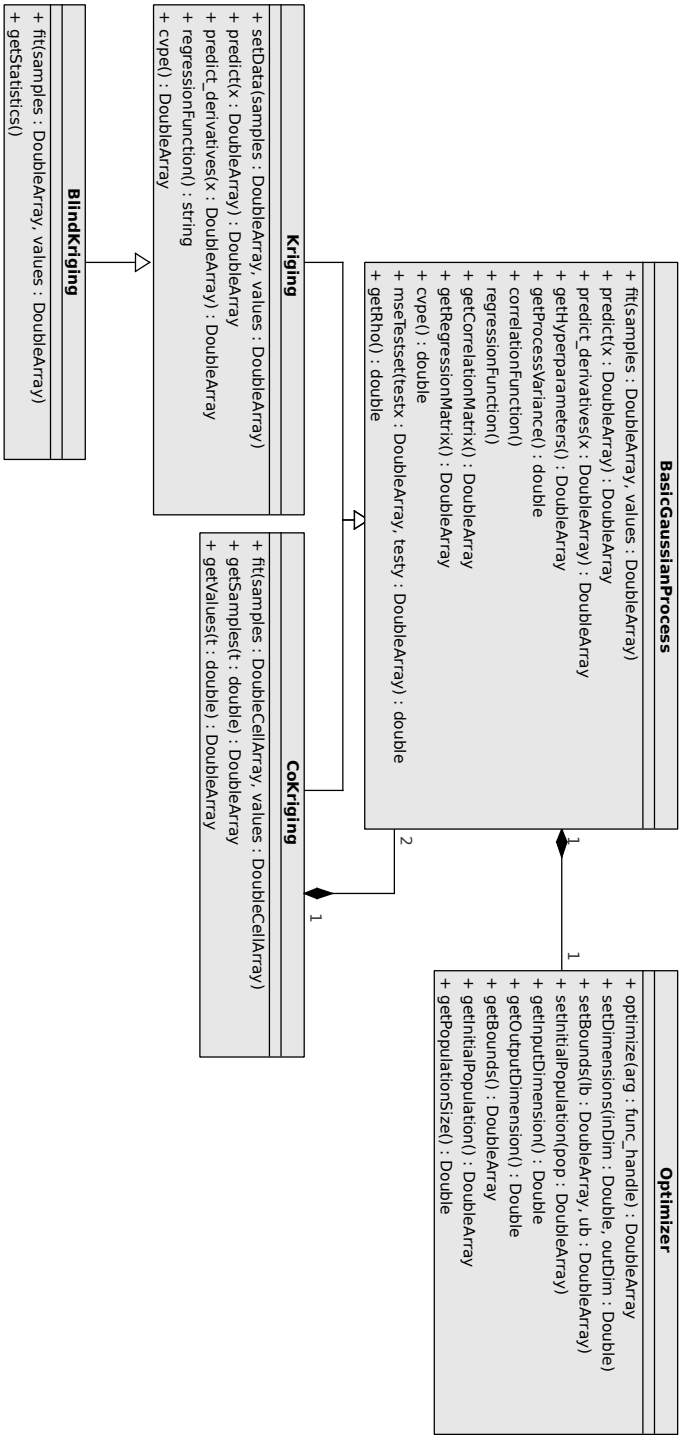


Figure B.3.: Class diagram of the ooDACE toolbox.

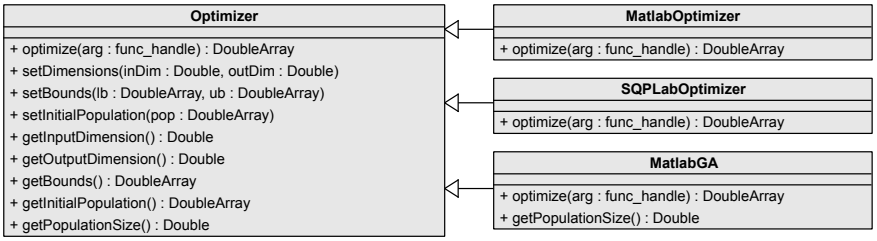


Figure B.4.: Class diagram of the Optimizer class hierarchy.

The remainder of this Section presents pseudo-code how to use the ooDACE toolbox for different use cases and types of Kriging models. See the included *demo.m* and *oodacefit.m* scripts for more example code on how to use the ooDACE toolbox, including more advanced features such as using blind Kriging (see the *BlindKriging* class) or how to use regression instead of interpolation. For convenience, wrapper scripts (*dacefit.m*, *predictor.m*) are provided that emulate the DACE toolbox interface (see Section B.2.5 for more information).

B.2.2.1. Kriging

Creating and exploiting a Kriging model requires only two lines of code:

```
k = oodacefit( samples, values );
y = k.predict(x);
```

For more flexibility, e.g., choosing the correlation and regression functions, the user can utilize the *Kriging* classes directly. **lb** and **ub** are $1 \times d$ arrays defining the lower bounds and upper bounds, respectively, needed to optimize the **hyperparameters**. In addition, a set of starting values has to be specified, namely, **hyperparameters0** is also an $1 \times d$ array. Example code to create an universal Kriging model follows:

```
...
% Generate Kriging options structure
opts = Kriging.getDefaultOptions();
opts.hpBounds = [lb ; ub]; % hyperparameter
    optimization bounds
% configure the SQPLab optimization algorithm (
    included)
opts.hpOptimizer = SQPLabOptimizer( inDim, 1 );
% create and fit the Kriging model
k = Kriging( opts, hyperparameters0, 'regpoly2',
    @corrgauss );
k = k.fit( samples, values );
```

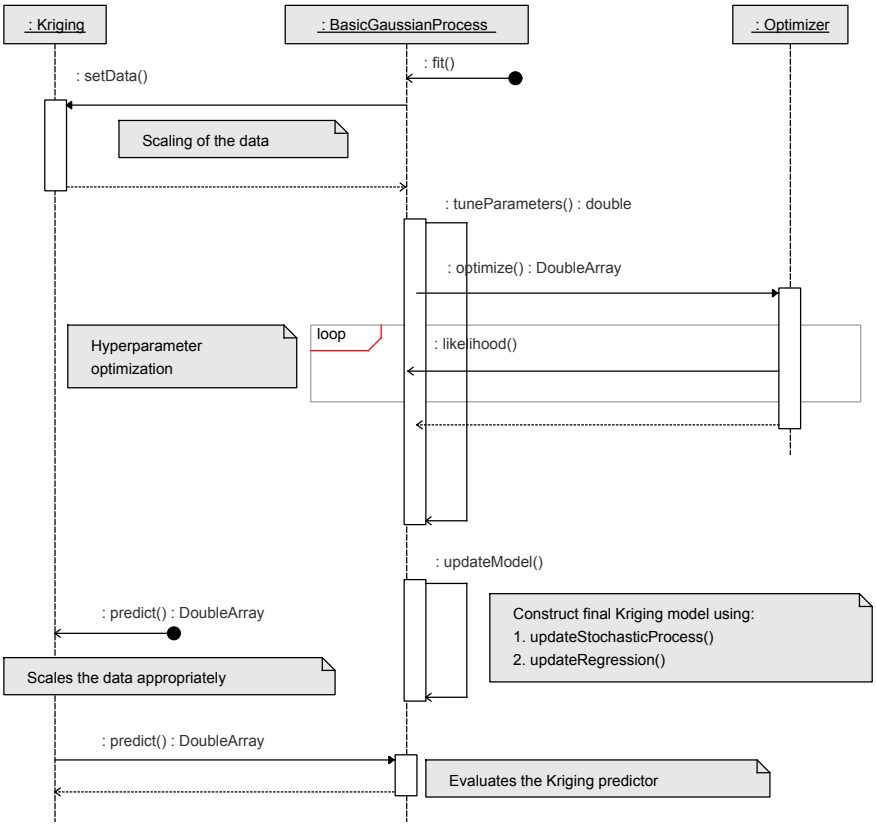


Figure B.5.: Sequence diagram of constructing a Kriging class.

```
k = k.cleanup(); % optional: only needed for very
    large datasets
% k represents the approximation and can now be used,
    e.g.,
[y mse] = k.predict( [1 2] )
...
```

The optional call to *cleanup()* after fitting a Kriging model clears some temporary, unused variables from memory to reduce memory usage, this may be especially useful for large datasets. Once a Kriging model has been constructed subsequent calls to the fit method of Kriging will use the previously optimized hyperparameters on the new data. This is useful for, e.g., calculating a 20-fold cross-validation score for which the hyperparameters need to remain fixed, some Matlab pseudo-code follows,

```
...
for i=1:20
  % fit cross-validated Kriging model
  k_xval = k.fit( samples(fold{i},:), values(fold{i
      },:) );
  % calculate error for this fold
  xval(i,:) = mean( (k_xval.predict( samples(fold{i
      },:) ) - ...
                    values(fold{i},:)).^2 );
end
% final score is the mean of all fold errors
xvalScore = mean( xval );
```

Here *fold{i}* are indices to a subset of the dataset for fold i. Note that leave-one-out crossvalidation (using the mean square error function) can be obtained directly using *k.cvpe()*, see Section 2.2.2.5.

B.2.2.2. Co-Kriging

The *oodace* script automatically creates a co-Kriging model if **samples** and **values** are cell arrays of length two. The first elements of **samples** and **values** describe the cheap data, represented by a $n_c \times d$ matrix (*samples{1}*) and a $n_c \times 1$ matrix (*values{1}*). Similarly, the second entries of both cell arrays contain the expensive data. A co-Kriging model is then created by executing:

```
k = oodacefit( samples, values );
y = k.predict(x);
```

On the other hand, the exact optimization strategies to use and other options can be defined when constructing the *CoKriging* class directly,

```
...
```

```
% Generate CoKriging options structure
opts = CoKriging.getDefaultOptions();
opts.hpBounds = [lb ; ub]; % hyperparameter
    optimization bounds
%% configure the optimization algorithms for
% for cheap data
opts.hpOptimizer{1} = SQPLabOptimizer( dim, 1 );
% for expensive data
optimopts.DerivativeCheck = 'off';
optimopts.Diagnostics = 'off';
optimopts.Algorithm = 'active-set';
optimopts.MaxFunEvals = 1000000;
optimopts.MaxIter = 500;
optimopts.GradObj = 'off';
opts.hpOptimizer{2} = MatlabOptimizer( inDim, 1,
    optimopts );
%% create and fit the CoKriging model
k = CoKriging( opts, hyperparameters0, 'regpoly0',
    @correxp );
k = k.fit( samples, values );
% k represents the approximation and can now be used,
    e.g.,
[y mse] = k.predict( [1 2] )
...
```

The co-Kriging model can be efficiently updated with new expensive data, which involves a re-estimation of the hyperparameters of one of the underlying Kriging models of co-Kriging. This is in contrast to Kriging were subsequent calls to *fit()* keep the hyperparameters fixed. When new expensive data arrives it suffices to update the second entry of the cell arrays samples and values with the new data,

```
...
samples{2} = [samples{2}; samples_new];
values{2} = [values{2}; values_new];
k = k.fit( samples, values );
```

Note that the co-Kriging class does not scale the data due to this feature as that might produce undesired results. It is suggested to scale (or normalize) your data to, e.g., $[0, 1]$, manually before calling the fit method.

B.2.2.3. Blind Kriging

A blind Kriging model is created using:

```
opts.type = 'BlindKriging';
k = oodacefit( samples, values, opts );
```

```
y = k.predict(x);
```

Additional options are,

```
% retune parameters after every iteration
opts.retuneParameters = false;
% maximum order of candidate features to consider (
    quadratic)
opts.regressionMaxOrder = 2;
```

Similarly to the previous Sections, a blind Kriging model can also be constructed directly by calling the constructor and fit method of the *BlindKriging* class.

B.2.2.4. Stochastic Kriging

For stochastic simulation problems a stochastic Kriging can be fitted using the *BasicGaussianProcess* class with the following additional options,

```
% Sigma is the intrinsic covariance matrix (=variance
    of output values)
opts.Sigma = var(values,2);
values = mean(values,2);
% the process variance sigma2 needs to be included in
    the MLE
opts.sigma20 = Inf; % optional, guess the initial
    value
opts.sigma2Bounds = [-2 ; 4]; % log10 scale
opts.generateHyperparameters0 = true; % optional,
    guess the initial value
% explicitly ask for BasicGaussianProcess (=Kriging
    without scaling)
opts.type = 'BasicGaussianProcess';
k = oodacefit( samples, values, opts );
[y s2] = k.predict(x);
```

On the other hand, when dealing with noisy simulators that are actually deterministic, one can use regression Kriging which tries to identify the amount of noise automatically by including an extra parameter λ in the likelihood optimization. A regression Kriging model is constructed using the *Kriging* class as follows:

```
% regression Kriging
opts.lambda0 = 0;
opts.lambdaBounds = [-5 ; 5]; % log scale
k = oodacefit( samples, values, opts );
[y s2] = k.predict(x);
```

As regression Kriging approximates the data, in contrast to interpolation, the prediction variance will not be zero at the samples. By specifying the option,

```
opts.reinterpolation = true;
```

before fitting the regression Kriging model, *predict()* will return the re-interpolated prediction variance as its second output argument.

B.2.3. Running the problems provided with ooDACE (demo.m)

The *demo.m* script includes several test cases trying to cover the most important aspects of the ooDACE toolbox. To solve a problem just run the demo script and make your selection of the several test cases, or, you can execute a test case directly by calling, e.g.,

```
demo(4)
```

Each test case creates a landscape plot of the Kriging model, as well as two contour plots of the prediction and the prediction variance (and the derivatives), respectively. These plots are found in Figure B.6. A quick discussion of the test cases follows.

B.2.3.1. demo(1) - fitting a standard Kriging model

This test case fits a standard ordinary Kriging model on the Branin function. This is the most simple use of the ooDACE toolbox and is probably the setup most users want to utilize. The output of Matlab is,

```
>> demo(1)

Building model...done

Evaluating model at (2.500000,7.500000).
Prediction mean = 31.627632.
Prediction variance = 24.001527.
Derivatives of: prediction mean = (5.887667,10.308727)
   . prediction variance = (0.001800,-0.000415).
Leave-one-out crossvalidation: 199.672399 (using the
   mean squared error function).
Integrated Mean Square Error: 23885.293094.
Marginal likelihood (-log): -6.586541.
Pseudo likelihood (-log): -5.778266.
Process variance: 23144.970800
Sigma(1,1): 0.000000 (first element of intrinsic
   covariance matrix).
```

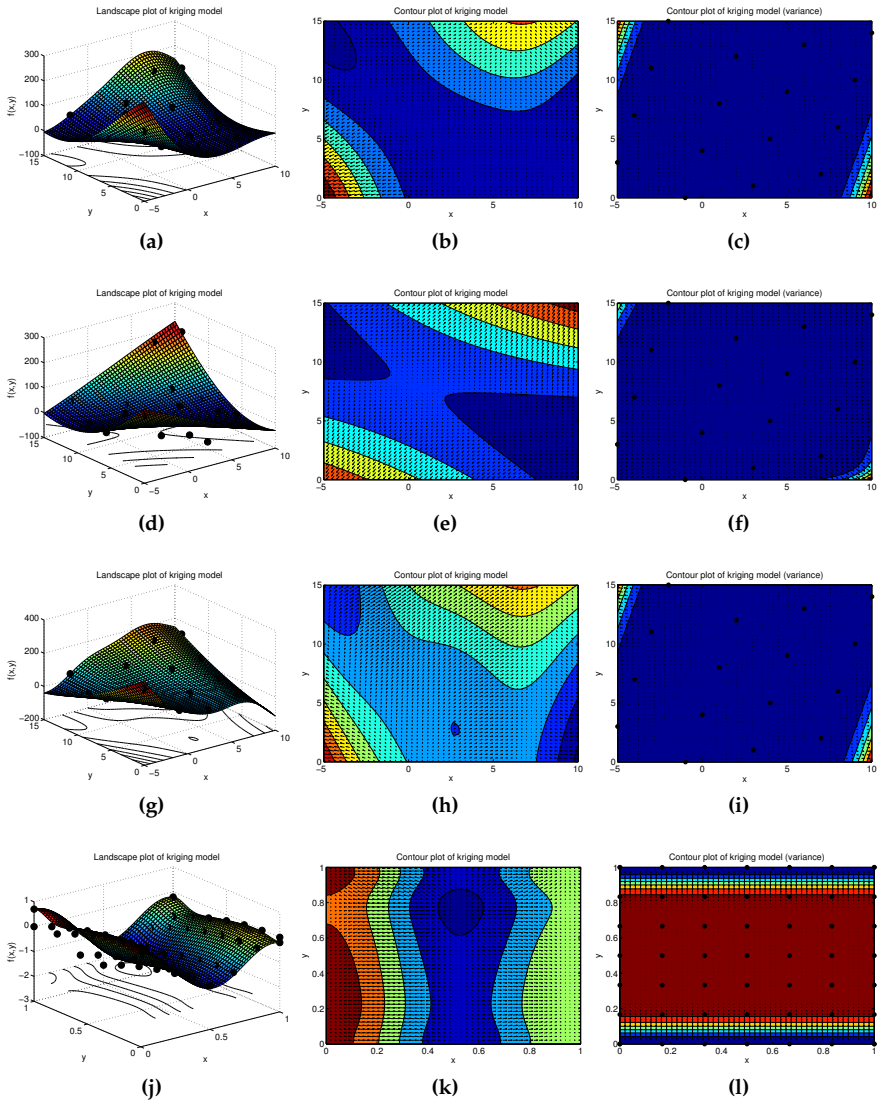
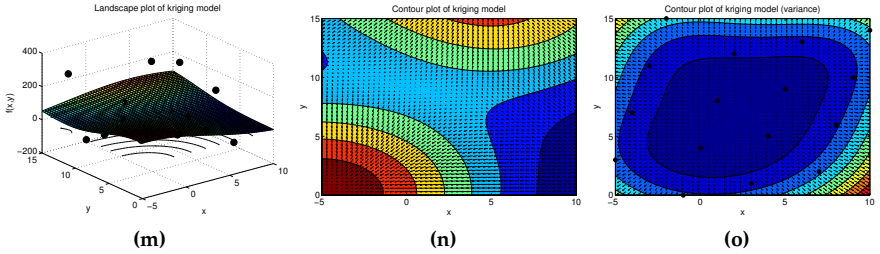

Landscape plot of kriging model
Contour plot of kriging model
Contour plot of kriging model (variance)
f(x,y)
x
y
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(l)

Figure B.6.



(m) (n) (o)

Figure B.6.: Plots generated by the several test cases of *demo.m*. The first row (a, b, c) is of the first test case (demo(1)), The second row (d, e, f) of the second test case (demo(2)), etc. The black dots are the samples and the black arrows represents the derivatives.

```
Formatted regression function: 0
Calculating derivatives for contour plot... (may take
    a while).

ans =

Kriging model with correlation function corrgauss (
    -0.28 -0.95 )
Average Sigma 5.773e-15
```

Beside the prediction and prediction variance (and their derivatives), several accuracy metrics are available giving an indication of the accuracy of the Kriging model.

B.2.3.2. demo(2) - fitting a regression Kriging model

This test cases demonstrate how to create a Kriging model for noisy data, we reuse the Branin function modified by adding some noise. We use the pseudo-likelihood to optimize the hyperparameters instead of the standard marginal likelihood and enable the re-interpolation of the prediction variance. This last option makes sure the prediction variance is zero at the samples, which is sometimes desired (e.g., for optimization). In addition, debug mode is enabled and, hence, a debug contour plot of the likelihood surface is calculated, see Figure B.7.

The output looks like,

```
>> demo(2)
Building model...
... (removed the output of creating the likelihood
    plot)
done
```

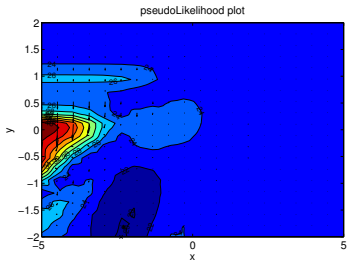


Figure B.7.: Contour plot of the marginal likelihood function for demo(2). The green cross are the optimal values found by the Maximum Likelihood Estimation (MLE), while the green star are the minimum values identified while generating this contour plot. For this particular setup x and y represent the hyperparameters σ^2 and θ_1, respectively. θ_2 is set fixed to the optimal value found by the MLE.

```
Evaluating model at (2.500000,7.500000).
Prediction mean = 0.655786. Prediction variance =
    0.000000.
Derivatives of: prediction mean = (-0.193567,3.153803)
    . prediction variance = (-0.001070,0.000983).
Leave-one-out crossvalidation: 4628.973432 (using the
    mean squared error function).
Integrated Mean Square Error: 2.059858.
Marginal likelihood (-log): 3.446575.
Pseudo likelihood (-log): 19.679406.
Process variance: 533479.248026
Sigma(1,1): 0.005171 (first element of intrinsic
    covariance matrix).
Formatted regression function: 0
Calculating derivatives for contour plot... (may take
    a while).

ans = Kriging model with correlation function
    corrgauss ( -1.82 -1.44 )
Average Sigma 5.171e-03
```

Note that Sigma(1,1) (= lambda hyperparameter) is larger than zero as we are doing regression instead of interpolation now.

B.2.3.3. demo(3) - fitting a blind Kriging model

This test case is the same as demo(1), except we are fitting a blind Kriging model. This type of Kriging tries to automatically determine the right regression (trend) function of the data. The output is,

```
>> demo(3)

Building model...done

Evaluating model at (2.500000,7.500000).
Prediction mean = 22.183247. Prediction variance =
   30.181832.
Derivatives of: prediction mean =
   (21.352608,27.792254). prediction variance =
   (-0.020281,-0.023028).
Leave-one-out crossvalidation: 34.381536 (using the
   mean squared error function).
Integrated Mean Square Error: 37946.333634.
Marginal likelihood (-log): -12.348891.
Pseudo likelihood (-log): -18.519979.
Process variance: 14881.714360
Sigma(1,1): 0.000000 (first element of intrinsic
   covariance matrix).
Formatted regression function: 1+x1^2+x1+x1x2
Calculating derivatives for contour plot... (may take
   a while).

ans = Kriging model with correlation function
   corrgauss ( -0.13 -1.21 )
Average Sigma 5.773e-15
```

As we can see there is a dramatic improvement on the leave-one-out crossvalidation score in comparison to the first test case. The final regression function, without the coefficients, is shown after 'Formatted regression function', namely, $1 + x_1^2 + x_1 + x_1x_2$.

B.2.3.4. demo(4) - fitting a co-Kriging model

This test case deals with multi-fidelity data, namely, two datasets modeling the same problem but coming from two different simulators with varying accuracy. Typical we have a dataset from an expensive and a cheap simulator. This data can be combined to enhance accuracy by creating a co-Kriging model. For demonstration purposes two mathematical functions are used here to represent the two simulators. The output looks like,

```
>> demo(4)
```

```
Building model...done

Evaluating model at (0.500000,0.500000).
Prediction mean = -1.373255.
Prediction variance = 0.136490.
Derivatives of: prediction mean =
   (-0.915713,-0.914880). prediction variance =
   (0.000000,0.000000).
Leave-one-out crossvalidation: 0.000004 (using the
   mean squared error function).
Integrated Mean Square Error: 0.110793.
Marginal likelihood (-log): -222.002058.
Pseudo likelihood (-log): 24557.664943.
Process variance: 280.984898
Sigma(1,1): 0.000000 (first element of intrinsic
   covariance matrix).
Formatted regression function: Not available
Rho: 0.915509
Calculating derivatives for contour plot... (may take
   a while).

ans =

Kriging model with correlation function corrgauss (
   0.08 -1.75 )
Average Sigma 4.058e-12
```

B.2.3.5. demo(5) - fitting a stochastic Kriging model

Finally, this test case demonstrates the stochastic Kriging model. This approximation model is used when dealing with data from stochastic simulations. Often the stochastic simulator provides error bounds on the output noise and/or multiple simulation runs are done to get an estimate of the amount of noise. Stochastic Kriging can use this extra information to improve accuracy. Here we use data from the Branin function with some random noise added to it. The output is,

```
>> demo(5)

Building model...done

Evaluating model at (2.500000,7.500000).
Prediction mean = 47.763651. Prediction variance =
   9999.999918.
```

```
Derivatives of: prediction mean =
   (-0.000280,-0.000112). prediction variance =
   (-0.001030,0.000924).
Leave-one-out crossvalidation: 20087.314366 (using the
    mean squared error function).
Integrated Mean Square Error: 2249999.982771.
Marginal likelihood (-log): 102.562327.
Pseudo likelihood (-log): 164.352336.
Process variance: 10000.000000
Sigma(1,1): 6843.391500 (first element of intrinsic
   covariance matrix).
Formatted regression function: 0
Calculating derivatives for contour plot... (may take
   a while).

ans = Kriging model with correlation function
   corrgauss ( -1.97 -1.97 )
Average Sigma 1.016e+04
```

The difference between stochastic Kriging and regression Kriging is that here the matrix Sigma is not included in the maximum likelihood estimation, but is defined a priori by the user (often by replicating the stochastic simulations a number of times).

B.2.4. Regression tests

The ooDACE toolbox also includes a regression test suite which can be run as follows:

```
runRegressionTests
```

Results of each test are compared against previous saved results (in regressionTests/) and when there are no problems found the output should be,

```
Running test 1...OK.
Running test 2...OK.
Running test 3...OK.
Running test 4...OK.
Running test 5...OK.
```

However, this will likely not be the case for most users as the regression tests are very strict. Small changes between Matlab versions (e.g., *dblquad* versus *integral2* for the *imse* method) and the Optimization toolbox (*fmincon*) will result in failed tests. Regression tests are most useful for developers: Before making changes to the code the output of the regression tests (for a particular Matlab setup) can be saved using *runRegressionTests([1:5],*

true);. After introducing new functionality to the ooDACE toolbox running *runRegressionTests* again will find any regressions in the code.

B.2.5. DACE toolbox interface

The ooDACE toolbox provides two scripts *dacefit.m* and *predictor.m* that emulate the behavior of the DACE toolbox [2]. Note, that full compatibility is not provided. The scripts merely aim to ease the transition from the DACE toolbox to the ooDACE toolbox. Example code,

```
krige = dacefit(samples, values, 'regpoly0', '
   corrgauss', hyperparameters0, lb, ub )
y = predictor([1 2], krige)
```

Obviously, a lot less code is used to copy the setups described in the previous sections. However, less code means less flexibility (e.g., blind Kriging and regression Kriging are not available using the wrapper scripts). Hence, it is suggested to learn the object oriented interface of ooDACE and use it instead.

B.2.6. Contribute

Suggestions on how to improve the ooDACE toolbox are always welcome. For more information please see the feedback page at `http://sumowiki.intec.ugent.be/index.php/Feedback`.

B.2.7. Bibliography

[1] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2006.

[2] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.